Examples

Year after year, there are fewer and fewer examples that boil down to a simple linear regression: they are replaced by non-linear (sometimes non-parametric) and/or hierarchical linear (or survival, logictic, Markov, spatial, even spatio-temporal multiscale (country/micro-region/region)) penalized models, sometimes with added boosting or a bayesian prior; some presenters use the bootstrap to have more robust confidence intervals.

Examples included

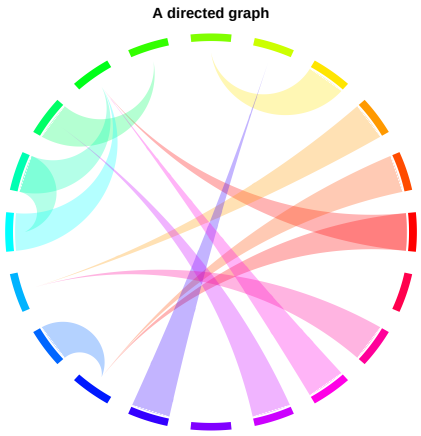
- Non-linear models in biochemistry, to estimate drug interactions, dose-response curves;
- Shape analysis;
- Text mining (for gene names in biology);
- Meta-analysis (aggregating several studies, e.g., with microarray data);
- Time series (building a business cycle (recession) indicator from NBER data: transform and/or filter the data, use non-linear and/or Markov switching (MS) models);
- Forecasting multivariate time series (to optimize a frozen goods supply chain or forecast electricity consumption);
- Multivariate analysis (with `FactoMineR`);
- Design of experiments (DOE), survey design (compute the sample size, in each stratum, for a given desired precision, to estimate fisheries revenues)
- Subject randomization system (subjects for a randomized trial arrive one by one; we then learn in which strata (gender, age, etc.) they are; we decide to give them treatment A or B by flipping a coin; we can hasten the convergence by keeping a running difference $n_A - n_B$ for each stratum and using a biased coin to redress any unbalance).

The large-scale ThomasCook example was interesting:

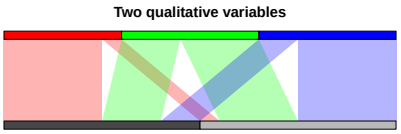
- They store the data in an SQLite table (modified to hold 30,000 columns – with PL/R, they had ended up writing R code that called PostgreSQL that called R: it was a nightmare to debug);
- They use R for ETL (Extract, Transform, Load), with some Python (BeautifulSoup) for screenscrapping;
- `glmpath` to select variables;
- `randomForest` (on the selected variables) to predict the speed at which the planes will fill;
- The prices are optimized with the random forest model;
- They add a fuzzy inference engine (based on the `sets` package) to prevent canibalization (you lose clients to yourself) and allow the business to add their imput to the model;
- They provide a graphical interface with wxPython/RPy2 (and py2exe to create Windows executables that do not require Python on the clients); R remained on a central server (some call it SaaS: Software as a Service).

Plots

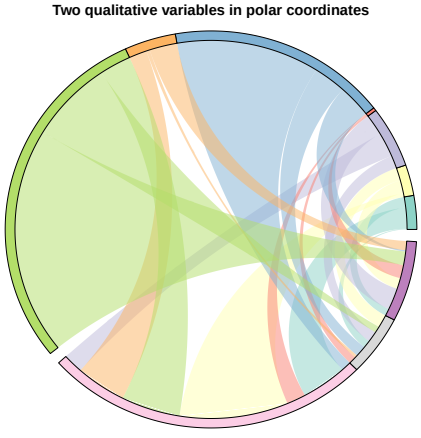
Kaleidoscope plots are those impressive, dense, polar coordinate plots, judiciously using transparent colours, you sometimes see in bioinformatics, often produced by Circos (GPL, unrelated to R). There was one on the title page of *Beautiful code, Compelling Evidence* (which explains how to use Haskell and OpenGL to produce graphics). The basic idea stems from the graph layout problem: if you are completely clueless about how to plot a graph, just put the vertices in a circle; if the graph is weighted, draw the edges as varying-width ribbons;



finally remark that a contingency table is a (bipartite: rows and columns) weighted graph

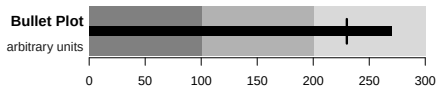


and draw it in polar coordinates (pay attention to the order of the segments and the ribbons, label the segments, add axes). Use it if you want artistic plots (that make the user wonder what the plot is about) or if your data is too large to be amenable to more traditional plots.

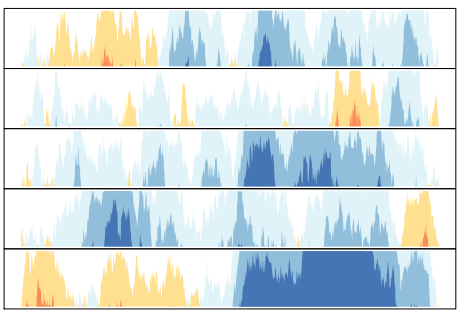


Visual analytics are statistical graphics made by artists or good communicators: they should capture the audience's attention and convey the speaker's ideas. The most salient such example is GapMinder (a flash-based application to display multivariate time series, as animated bubble plots) and its narrative power (it helps you efficiently *tell* a story, but might not be the best tool to *discover* that story). As statisticians, we can try to reduce the information displayed to what is relevant and add model information (distribution, prior, expected values under H_0, etc., for instance a map of cholera cases can be complemented by well positions and their Voronoi tessellation). R provides graphics for diagnostics, presentation (but no animations or interactivity), and not much for data exploration (iplot, iplot eXtreme, ggobi) – redundant functions do not help (for instance, you can draw a histogram in any of the following ways: `hist`, `truehist`, `histogram`, `qplot(...,geom="histogram")`, `ihist`).

An **information dashboard** (ID) presents all the information needed in a single, easy-to-read page; it can use bargraphs, stacked bargraphs, linegraphs, scatterplots, boxplots, sparklines, treemaps, **bullet graphs** (a thick horizontal line from the origin to represent a single value, with a tick for a reference point (target, last year's value, etc.), and a coloured background to represent how good the value is (3 values, from dark (bad) to light (good) grey; do not rely on colours); do not forget a textual label, with units (smaller), and a scale)



and **horizon plots** (to plot a time series, represent both positive and negative values above the axis, with a different color (blue/red), put the possible values into three bands, of darker colours, and overlay them; if the slopes are around 45 degrees, you can easily compare the slopes in the positive and negative regions and check if the variance changes) – of course, no piecharts.



The `iPlot` package adds **interactive graphics** to R, but cannot be used for large datasets: there is no OpenGL in Java and the data is duplicated between R and the JVM. The soon-to-be-released iPlot eXtreme, in C++, should address those problems.

The `PairViz` package uses `graph` and `igraph` to order the variables in a **parallel coordinate plot** using graph-based algorithms (hamiltonial or eulerian path); this can be combined with **scagnostics** if there are many variables.

R can be linked to external visual exploratory analysis tool: ggobi, Visplore (opensource, but early alpha), etc.

To plot the results of a clustering,

```
bwplot( value ~ variable | cluster )
```

you can add a reference point (the global average of each variable), fine-tune the order of the clusters, of the variables, highlight important variables (or grey out unimportant ones – this will depend on the cluster).

Statistical Tests

Multiple testing often assumes the tests are independent and then controls the *family-wise error rate* (FWER, probability of having at least one wrongly rejected null hypothesis), the *false discovery rate* (FDR, proportion of wrongly rejected null hypotheses) or the *false non-discovery rate* (FNDR, proportion of wrongly non-rejected null hypotheses). If you simulate data similar to yours, you can have an idea of the distribution of the FDR and FNDR and see how it changes as the correlation of the data increases. It is possible to include some knowledge of the corelation structure into the testing procedure: these "factor-adjusted test statistics", implemented in the `FAMT` package (factor analysis for multiple testing), can improve both FDR and NDR.

When estimating a p-**value** by Monte Carlo simulations (bootstrap, permutations, etc.), some people do not want the p-value but are happy with just knowing if $p \leqslant \alpha$ for some predefined threshold α. In this case, we can do with slightly less data (than if we wanted p, *i.e.*, all values of α) by looking at the path $(\sum_{1\leqslant i\leqslant n} T_i)_{n\geqslant 1}$ of partial sums of the test statistic and checking if it remains in or leaves some box- or tunnel-shaped region (sequential Monte Carlo p-values).

The `car::Anova` function implements nested **linear tests** ("type II" tests, *i.e.*, tests of $H_1|H_2$, where $H_1 \implies H_2$) and unconditional linear tests ("type III" tests) with an F statistic.

Matching

When comparing the two models $y \sim x$ and $y \sim$ treatment$+x$, where the treatment is a binary variable, some people like to have two observations for each possible value of the covariates, one for each value of the treatment. This is usually not the case, but we can instead "fabricate" new observations: **matching** is very similar to regression, but it is not linear (it could be seen as a "non-parametric regression developed by non-statistitians"). This also looks like an imputation (or inference with missing data) problem, but it is apparently addressed with ad hoc methods: coarsened exact matching (CEM) is just one of them – there are 10

matching packages for R... An application to program evaluation ("program" means sanitation improvement, etc.) was presented.

Penalized Estimators

Penalized estimators (such as non-parametric regression) are good but biased by construction: you can try to estimate and reduce the bias; the `ibr` package implements such an **iterative bias reduction** – but there were too few details in the presentation.

Penalized regression minimizes RSS + $\sigma^2 k\lambda$, where RSS is a sum of squares, k is the model size, and λ the penalty for complicated models. AIC (Akaike information criterion) uses $\lambda = 2$ but overfits large datasets; BIC (bayesian information criterion) uses $\lambda = \log n$. **Adaptively penalized regression** minimizes RSS + $\sigma^2\phi(k, n)$, where ϕ is motivated by the FDR (false discovery rate), *i.e.*, the expected number of wrongly rejected H_0 in a multiple test (it has a higher power than the too conservative FWER (family-wise error rate), which controls the probability of one or more incorrectly rejected H_0; with the FDR, you do not have a single p-value, but a *sequence* (for the first, second, etc. test): p/m, $2q/m$, $3q/m$).

A full **regularization path** (*i.e.*, a penalized estimator, not for one for all values of λ: lasso, with its L^1 penalty; ridge, with its L^2 penalty; elastic net, with both: $\frac{1}{2}(1-\alpha)\left|\beta\right|^2 + \alpha\left|\beta\right|$) can be efficiently computed by **coordinate descent** (discretize the path, estimate one coordinate at a time until convergence, repeat for the next point on the path, etc.). Coordinate descent should work any time your optimization if of the form something$+\lambda\cdot$penalty, for instance to build undirected graphical models (it gives you a sequence of graphs) or to complete a matrix (e.g., a movies$\times$rates matrix, not observed completely).

In **credit rating**, to estimate the probability of default, you can replace the (linear) logit model by a generalized additive model (GAM). (There are (too) many such estimators in R, not only `gam` and `mgcv`; if you hesitate, use `gam` (more stable for small datasets), unless you have a lot of relatively clean data.)

The `Influence.ME` package provides regression diagnostics (influential observations, etc.) for mixed models; the `merBoot` package uses bootstrap to provide empirical p-values for mixed models.

Non-linear models

The `nlstools` package provides a few functions to help fit non-linear models: `preview` to help you (graphically) choose the starting values; `nls` to fit the data; `overview` as an alternative to `summary`; `plotfit` to display the data and the fitted curve – also check `nlsResiduals`, `test.nlsResiduals`, `nlsContourRSS` (to identify ill-conditioning), `nlsConfRegions`, `nlsJack`, `nlsBoot`.

Univariate data

To estimate the **mode** of unimodal data, you can compute density estimators with wider and wider kernels, look at their local maxima and remember their positions when they disappear: you get a dendogram-like plot – this can actually ge generalized to higher dimensional data (apply some dimension reduction algorithm afterwards, if needed): check the `denpro` and `delt` packages.

The `DTDA` package provides functions to analyze truncated data (estimator of the cumulated distribution function (cdf), etc.).

The `fitdistrplus` package helps you fit distributions (as `MASS::fitdistr`), but allows for *censored* observations, provides skewness/kurtosis plots (with the dataset, bootstrap replications, and the regions attainable by the models you are considering) and goodness-of-fit tests (Anderson-Darling test to compare the data and the fitted model – it accounts for estimated parameters).

The Tobit model

$$\begin{aligned} y_1 &= \beta x + \varepsilon \\ y &= y_1 \cdot 1_{y_1>0} \end{aligned}$$

can be generalized to **multiple hurdle** models

$$\begin{aligned} \mathrm{y} &= \beta \mathrm{x} + \varepsilon \\ \varepsilon &\sim N(0, V) \\ y &= y_1 \cdot 1_{y_1>0, y_2>0, \ldots, y_m>0} \end{aligned}$$

with the `mhurdle` package.

The `mlogit` package fits multinomial logit models (wasn't there already an implementation in `nnet::multinom`?).

Time Series

Markov Switching models can be fitted by expectation maximization (EM): estimate the state at each time; then the parameters in each state; iterate.

You can **mine** a large number of time series by computing a few metrics for each of them (average, volatility, minimum and maximum of the derivative, etc. – use the methods of *functional data analysis* (FDA)) and plot the resulting multivariate dataset (1- or 2-dimensional plots, PCA (principal component analysis), etc.). This is the idea behind **scagnostics**, applied to time series.

In a *threshold autoregressive model* (TAR) model, the autogression (AR) coefficient depends on the previous value (often, one allows two values depending on the sign; sometimes three values corresponding to "significantly positive", "significantly negative", "small"). The idea can be generalized to **threshold cointegration**: two time series are threshold cointegrated if they have a threshold stationary linear combination y, *i.e.*, $y_{t+1} = \alpha(y_t)y_t + \varepsilon_{t+1}$, with $|\alpha(y_t)| < 1$ if $|y_t|$ is large.

The Kalman filter can be robustified to resist to either outliers or regime changes (or even both, but with a

delay) with the rLS and ACM algorithms (not detailed enough).

With the `seewave` package, R can process sound samples (the plots are nice and the PDF presentation even contained an animation) – of course, since R tends to store data in memory, this is only limited to sound *samples*.

Breakpoint models are used in bioinformatics to model the ratio

$$\frac{\text{number of copies of a gene}}{\text{number of copies in the reference}}$$

as a locally constant function of the position on the chromosome; the breakpoints can be estimated with *dynamic programming* – could dynamic programming be applied to other breakpoint problems? I rarely see it outside bioinformatics.

The usual RMetrics presentation(s) focused on date arithmetics (the `timeDate` and `timeSeries` packages are aware of weekdays, businessdays, for a given financial center) and *portfolio optimization*.

Spatial Models and GIS

The `GeoXp` package provides a few plots for spatial data: *driftmap* (not defined), *angle map*,

$$f(x1) - f(x2) \sim \text{angle}(x2 - x1),$$

where the angle is with respect to the horizontal axis), *neighbourhood map* (number of neighbours versus distance to the nearest neighbour), *Moran plot* (to check for spatial autocorrelation, plot the value of the variable of interest at a point versus the average value in a neighborhood of this point – you could also use two different variables), outliers (Mahalanobis distance between pairs versus Euclidian distance between pairs); those plots could be be linked to a map.

Multivariate Data

To the list of *multidimensional scaling* algorithms, `cmdscale`, `isoMDS`, `sammon`, you can add `isomap`, which tries to find a (non-linear) 2-dimensional submanifold, smooth but close to the data. As always, try all algorithms and try to replace the dissimilarities by their ranks (it does not matter if the matrix is not metric).

The `biclust` package implements many **biclustering** algorithms (for binary, quantitative, ordinal or qualitative variables).

Hierarchical clustering can be performed on principal components (some claim that discarding the last factors of a factor analysis makes the clustering more robust, others claim the opposite); the clusters can be defined by representative individuals (actual data points), variables (as in biclustering) or axes (as in a biplot).

fMRI data can be analyzed through spatial and even (spatio-)temporal ICA (independent component analysis) if you have a sparse description of the singular value decomposition (SVD); also check the `PTAk` package for principal tensor analysis – a similar idea was presented a few years ago at SigGraph.

Canonical correspondance analysis (CCA) and *partial least squares* (PLS) are very similar: the former looks for linear combinations $\alpha'X$ and $\beta'Y$ of X and Y that maximize $\text{Cor}(\alpha'X, \beta'Y)$, the latter maximizes $\text{Cov}(\alpha'X, \beta'Y)$. Both can be regularized: the ridge (replace $(XX')^{-1}$ with $(XX' - \lambda I)^{-1}$) is popular with CCA, the lasso (an L^1 penalty) with PLS.

Data envelopment analysis (DEA, *i.e.*, the construction of *efficient frontiers* in the input × output or risk × reward space) can use CCA as a first step, to select the variables to use; beware of outliers: DEA focuses on extremes. DEA can also be applied to medicine.

Distributed quantile estimation (compute a few quantiles on each client, convert them into an estimated cumulated distribution function (cdf), send it to the server which then averages those cdfs to estimate the quantiles) can be generalized to higher dimensions with PCA (only compute the quantiles on the first principal components); this idea can be applied to *monitoring* (e.g., to spot when a coffee machine is about to run out of coffee by listening to it).

Multiple tests and variable selection can be improved by accounting for correlation (with a correlation-adjusted T score). For instance, in genetics, you can group genes in the same pathway (or, if you do not have this information, in the same correlation neighbourhood). For feature selection, you should control the FNDR (false non-discovery rate) rather than the FDR (false discovery rate).

Several presentations focused on **multiple imputation**: exploring/visualizing the structure of missing values with the `VIM` package (there is also a GUI); fixing the separation problem (more variables than observations would remove all the randomness from the imputation procedure) by adding a bayesian prior with the `mi` package; reducing the bias in variance estimates after multiple imputation with a misspecified model with "estimating equations" (no details).

Trees

The `partykit` package unifies the various tree packages (`rpart`, `knnTree`, `Rweka`, `party`, `randomForest`, `gbm`, `mboost`) and should facilitate the implementation of other algorithms; it also provides a PMML interface (PMML is an XML file format to exchange statistical models between applications, mainly used in data mining).

Random forests are not only used for *prediction* or *feature selection*, but also to measure how important each variable is; contrary to univariate measures, they can take interactions into account. The **permutation importance** of X_j is the average decrease in classification accuracy after permuting X_j. This is problematic because a variable can appear important because it is in-

dependent from the variable to predict, or just from the other variables. Conditional permutation importance (implemented in the `party` package) can help focus on the former.

The standard benchmark to measure the performance of a tree algorithm is C4.5 – its licence does not allow you to use it, but it has been reimplemented as J4.8 in Weka, available through `RWeka`. Other algorithms include: CART (classification and regression trees) in the `rpart` package; unbiased recursive partitioning (QUEST in the `LohTools` package) and conditional inference trees (CTree in the `party` package). They all perform well, except C4.5/J4.8 without cross validation (cross-validation improves *all* those algorithms and is already included in some of them: if not, do not forget to add it).

Classification trees (CART, in the `rpart` package) can be generalized to a multivariate gaussian outcome (`longRPart`).

After a hierarchical classification, one usually cuts the dendogram, but you do not have to do it horizontally – *permutation tests* can help you find a good, non-horizontal cut.

Graphs

While the *graphical model* R packages landscape is getting less empty, it remains more heterogeneous than other domains, and is still plagued with external, Windows-only and/or closed source solutions, rarely useable in a professional context: WinBugs, MIM, Genie/Smile (closed-source, but free to use, even commercially), etc. Here are a few examples besides the classical `gR` (actually a combination of several packages) and `gRbase`:

- The `ReBaStaBa` package describes an already constructed bayesian network (*bayesian networks* provide a concise description of a joint distribution);
- The `IdR` (influence diagrams in R) package, can apparently learn the graph structure from the data – too few details in the presentation.

Graph (or link) data abound, from the web (links between Wikipedia articles) to linguistic databases. An **ontology** is a list of words, with links between them; contrary to a taxonomy, those relations are not limited to speciation/generalization but can be arbitrary complicated; polysemy and synonyms are often prohibited (to avoid the mess of gene and protein names); ontologies can be used to represent knowledge and make deductions (an ontology can be seen as a graph theorist's reinvention of Prolog). OWL (Web Ontology Language) may appear powerful, but is not robust at all; good old RDF (resource description language: it lists triplets (subject, predicate, object), or equivalently (node_1, edge type, node_2), *i.e.*, describes a graph with coloured edges – contrary to OWL, it does not allow for constraints on the predicates such as cardinality, transitivity) is sufficient for most practical purposes.

The `igraph` and `pagerank` packages provide graph-theoretic algorithms; for instance, they can be used to create a **tag cloud** using multidimensional scaling (MDS) for the positions. You may also want to check Wordle (non-free, but free to use), which takes the shape of the words into account, or Many Eyes's bubble chart (apparently no licence), which uses a circle packing algorithm and amounts to a square root transform.

The `sna` package (social networks) can be combined with the `tm` package (text mining) to analyze mailing lists.

The result of a centroid-based clustering algorithm (k-means, PAM (partition around medoids), etc.) can be assessed with:

- The *silhouette plot*, $d(x, c_2(x)) \sim d(x, c_1(x))$, where $c_1(x)$ and $c_2(x)$ are the centroids closest and second closest to obervation x;
- Topology-representing networks (TRN) or *neighbourhood graphs*, weighted graphs with the centroids as vertices and the weight between c_1 and c_2 is the number of points x such that $c_1(x) = c_1$ and $c_2(x) = c_2$ or the average mean average distance.

The `gcExplorer` package relies on `Rgraphviz`, `graph` and `symbols` (to add plots (barplots, etc.) in the graph vertices) to display these, interactively.

Machine Learning

Association rule mining can be seen as a form of "non-symmetric correspondance analysis" for binary data: it can rely on brute force or multiple correspondance analysis (MCA); some algorithms also use exogenous data (e.g., socio-demographic data for a client × item dataset).

The `set` package provides (finite) set operations, that can easily be generalized to **fuzzy sets** (just replace the topos of truth values $\{0, 1\}$ by the interval $[0, 1]$); applications include relations (order, partial order, equivalence, etc.) and how to average them (e.g., **voting systems**) – this leads to non trivial optimization problems.

The `TopkLists` package implements algorithms to average **ordinal data**, *i.e.*, rankings, when m raters provide the top k elements in a list of N (more voting systems).

Dynamical Systems

Surprisingly, R is sufficiently fast to simulate dynamical systems (e.g., in ecology), *i.e.*, large systems of differential equations; check the following packages: `desolve` (rather than `odesolve`, which could only tackle small/toy problems) for dynamical systems; `rootSolve` for the steady state solution; `linSolve` for least square solutions (when your linear system is undetermined, just add Min $\|x\|^2$ to turn it into a quadratic program); `simecol` for applications in ecology.

High Performance Computing

`bigmemory`, which uses the Boost C++ interprocess library, paliates the lack of multithreading in R through shared memory or file-based matrices; it also provides iterators to hide (or make implicit) the parallelism provided by `multicore`, `snow`, `nws`. Also check `ff` (matrices limited to $2^{31} - 1$ elements), `ffdf` (as fast as bigmemory), `sqlite` (sufficiently fast), `BufferedMatrix` (inefficient), `filehash` (sufficiently fast).

`RCpp` allows (eases) the use of C++ and STL containers; conversely, `RInside` allows you to call R from C++; also check `rdyncall` to use C libraries from R.

Several packages provide some parallelism: `multicore`; `snowfall`, `nws`, `Rmpi`; `gridR` (with parallel computations, beware of random number generators: the random numbers on each node should look independent).

Threads (as opposed to parallelism) are needed to process real-time (streaming) data. R will remain monothreaded for the foreseable future; in the meantime, use several process (with some C/C++ where needed) and shared memory (with bigmemory) between processes.

R can be used in simple web applications (the user fills in a form and a plot is updated), thanks to the `RApache` module, with some Ajax (Prototype, Scritaculous, YUI, jQuery, Dojo, Ext), XML or JSON (better than XML for large datasets), and `hwriter` (for static reports, I use `Sweave`).

`RdWeb` is a (still immature) web interface to parallel computing in R (`RApache` and `Rserve` do not focus on parallelism); it requires a batch system (`at`, openPBS, Torque, etc.)

A **task scheduler** is a hybrid between `make` and `at`/`crontab`: it automatically executes tasks after a given time, when some conditions (file availability, etc.) are met and allows tasks to be dependent on one another (you can write a quick and dirty task scheduler with `make` and `crontab`: from the crontab, launch `make -k -j 100` in a directory every minute; tasks simply fail if their starting conditions are not met; track which tasks have run or are currently running by creating empty files with predefined names). The Coalition task manager (in Python, with Twisted Matrix, on GoogleCode) handles dependencies, load balancing; it does not seem to provide repeated tasks ("every day"), or tasks starting after a given time. There is nothing specific to R (and there should not be): it just launches executables, which can be R scripts.

A **workflow engine** allows you to write (the equivalent of) Makefiles (*i.e.*, describe dependencies between tasks) graphically and run them on a grid, such as Knime, which focuses on *data pipelines*: it provides all the operations you can want on tables of data, the most common data mining algorithms and plots (Weka), and interfaces with R and BIRT (a J2EE reporting system).

The Nimrod toolkit combines a workflow system (Kepler), a design of experiments framework (DOE, more precisely "computer experiments", whatever that means – simulations?) and **gaussian processes** (with the `mlegp` package).

Biocep-R is a javaesque application server: huge, exhaustive, unwieldy and fully buzzword-compliant.

The `hive` package uses Hadoop (a distributed computed framework, with a distributed file system, trying to run the computations near the data, initialy developped by Yahoo and currently maintained by the Apache foundation) for text mining tasks (with the `tm` package).

Most of those cloud computing frameworks (Amazon EC2, etc.) can generate or use models in the PMML format (designed to exchange statistical models between applications).

GUI

JEdit can be used as an IDE for R (many people advise **Eclipse/StatET**): it interfaces with the `codetools` package (warnings are presented as spelling mistakes would be in a text editor); provides a tree view of the code (based on the R parser), completion popups (for colours or plotting symbols, you have the actual colour or symbol), an object browser, a debugger.

The building blocks for SciViews (an R GUI) are available as individual, reuseable packages; some were used to build other GUIs such as Zoo/PhytoImage, an R-based graphical software for image analysis; it also uses the ImageJ image processing Java library (public domain).

Mayday is a visualization platform (in Java, interactive, aware of meta data, with the ability to write plugins) for numeric matrices; it now has an interactive R shell, based on RLink, a replacement (or a layer above) rJava.

Windows

Many mentions of `rcom` and `statconnDOM` (R–C# link) to interface R and Excel (there is even a book on the subject – frightening) or generate powerpoint files.

R Ecosystem

In many domains, R provides very similar functions in different packages each with its own syntax. Some (well needed but) isolated **unification** works have begun, for instance `Zelig` (statistical models), `partykit` (recursive partitioning algorithms) or `optimx` (for optimization: it provides a single interface and compares the various algorithms, to help the user choose the best for the problem at hand).

Instructional presentations included **sparse matrices** with the `Matrix` package (used for linear mixed models by `lme4`) and spatial autocorrelated (SAR) models; **S4 classes** (the dispatch is based on the type of all arguments, not just the first as with S3 classes); implementing new statistical distributions (the dpqr functions, and a few others), as exemplified in the `distr` and `VarianceGamma` packages, with some good general **software engineering** advice.

Provenance tracking tries to answer the question "where does this variable come from? how was it modified?"; it is implemented in CXXR (a C++ reimplementation of R, designed to easily produce experimental versions of the R interpreter): the `pedigree` function is similar to `history`, but the *audit trail* it returns only gives the commands that affected a given object. The Open Provenance model aims at provenance tracking across systems.

Reproducible computing can be easily obtained by recording ("blogging") everything you type, including outputs and plots, in a way reminiscent of Maple worksheets. Social data networks (Many Eyes, Swivel, thedata.org, etc.) are starting to appear (but they often present pure coincidences as hard evidence).

Residual-based shadings for visualizing (conditional) independence
A. Zeileis et al.

Mosaic plots (that display contingency tables) and *association plots* (that display the Pearson residuals of a log-linear model $r_{ij} = (n_{ij} - \hat{n}_{ij})/\sqrt{n_{ij}}$) can be augmented by shadings or colours to display the significance of (*i.e.*, the p-value of a statistical test for) departure from independence:

- The usual χ^2 of independence, $\chi^2 = \sum_{ij} r_{ij}^2$ can be replaced by $M = \text{Max}_{ij} |r_{ij}|$, so that the whole test is significant iff a cell is significant;
- The critical values of this test (used to decide whether to colour or not) are data-dependent: they can be computed by *permutation tests*;
- The traditional colourings rely on the HSV (hue, saturation, value) **colour space** (a cone, whose vertex is black, whose base has a fully saturated rim and a white center) and can be misleading (saturation is not uniform accross hues); it can be replaced by the **HCL** (hue, chroma, luminance) space (a double cone, with a white vertex on one side and a black vertex on the other, and a fully saturated rim in the middle – that rim is slightly broken: admissible chroma and luminance values depen on the hue);
- Instead of changing just the saturation, you can also change both chroma and luminance;
- When there is nothing significant, use a medium grey (not white); as significance increases, increase the range of hues (a more colorful picture looks more significant).

Those ideas are implemented in R in the `vcd` package: check the `mosaic`, `assoc` and `cotabplot` functions.

Multivariate copulas at work
M. Fischer
RMetrics Workshop 2009

While binomial copulas are galore, "higher dimensional" ones (dimension 3 or 4, not the 200 or 2000 I needed a few years ago) used to be rare. The situation is slowly changing:

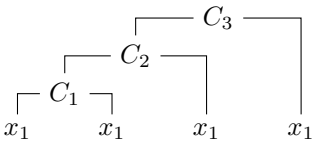
- Elliptical copulas;
- Archimedian copulas

  $$C(x_1,\dots,x_n)=\phi^{-1}(\phi(x_1)+\cdots+\phi(x_n))$$

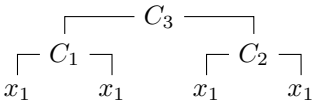
  which can be written multiplicatively (set $\theta(x)=\exp-\phi(x)$)

  $$C(x_1,\dots,x_n)=\theta^{-1}\left(\prod_i\theta(x_i)\right)$$

- **Generalized archimedian copulas**: use archimedian copulas to group variables two at a time, in a fully-nested way

  or using an arbitrary tree;

- **Generalized multiplicative archimedian copulas**: replace the independent copula $\prod$ by another copula C_0

  $$C(x_1,\dots,x_n)=\theta^{-1}(C_0(x_1,\dots,x_n));$$

  this can be seen as a ($\mathfrak{S}_n$-symmetric) deformation of C_0;
- **Liebscher copulas** combine several multiplicative archimedian copulas

  $$C(x_1,\dots,x_n)=\Psi\left(\frac{1}{m}\sum_j\prod_i\theta_j(u_i)\right);$$

- The **Koehler-Symanowski copulas** are build as follows: take n iid random variables

  $$Y_1,\dots,Y_n\sim\text{Exp}(\lambda=1);$$

  choose random variables to model bivariate associations

  $$G_{ij}\sim\Gamma(\alpha_{ij},1),\quad i<j;$$

  (and trivariate associations if you want: $G_{ijk}\sim\Gamma(\alpha_{ijk},1)$) and try to put them all in a formula, e.g.,

  $$U_i=\left(1+\frac{Y_i}{\sum_j G_{ij}}\right)^{\sum_j\alpha_{ij}}$$

- Starting from a (graphical model) decomposition

  $$f(x_1,x_2,x_3)=f(x_3)f(x_2|x_3)f(x_1|x_2,x_3)$$

  one can express a copula as a (complicated) product of bivariate copulas; this is the **pair copula decomposition** (PCD).

A maximum likelihood fit of those on several financial datasets suggests to forget plain archimedian copulas: elliptic (Student) or pair copulas provide a better fit.

Pair-copula constructions of multiple independence
K. Aas et al.

Detailed presentation of pair copulae. The decompositions are not described by a single graphical model but by a (maximal) nested set of (tree) graphical models – a D-vine. Imposing some conditional independence greatly simplifies the model.

Package development in R: Implementing GO-GARCH models
B. Pfaff
RMetrics Workshop 2009

The univariate GARCH model can be extended to a multivariate model by direct generalization (VEC, BEKK, factor models: keep the same formulas, with matrices instead of real numbers, and impose some conditions on them to keep the number of parameters reasonable); by linear combinations of univariate GARCH models (GO-GARCH (generalized orthogonal), latent factor GARCH) or even non-linear combinations (dynamic conditional correlation, general dynamic covariance models).

Consistent return and risk forecasting for portfolio optimization using kernel regressions
J. Hindebrand and T. Poddig
RMetrics Workshop 2009

Separately estimating expected returns and risk model (variance matrix) is suboptimal (when your only source of information is the time series of returns). The authors suggest to estimate both from past returns using *kernel regression* (apparently sometimes called "general regression neural network" (GRNN)) which can be seen as a smoothed k nearest neighbours algorithm. The risk can be estimated "explicitely", from the residuals

$$\hat\sigma^2=\frac{1}{k}\sum(\hat y_i-y_i)^2$$

or "implicitely"

$$\hat y=\frac{1}{k}\sum y_i$$
$$\hat\sigma_i=\frac{1}{k}\sum(\hat y-y_i)^2$$

(the sums are over the k nearest neighbours; for kernel regression, replace those averages by weighted averages).

Financial crises and the exchange rate volatility for Asian economies
T. Oga and W. Polasek
RMetrics Workshop 2009

Markov switching AR-GARCH models are amenable to bayesian MCMC estimation.

Markowitz and two managers
K. Rheinberger
RMetrics Workshop 2009

In many companies, portfolio managers work independently and their strategies or portfolios are combined by higher authorities (*i.e.*, they decide on the capital allocated to each). This is suboptimal: if the asset classes are correlated, the optimal global portfolio need not be obtainable as a combination of efficient subportfolios.

Simple parallel computing in R with Hadoop
S. TheuSSl
RMetrics Workshop 2009

MapReduce (also called **Cloud Computing** or **SAAS** (software as a service)) is a trendy name for "distributed divide and conqueer" where the task division step is aware of the location of the data on the *distributed file system* (DFS) to limit network usage. R can use *Hadoop*, a (Java) MapReduce implementation, initially developped by Yahoo and now maintained by the Apache foundation.

Modeling and evaluation of insurance risk using `actuar`
V. Goulet
RMetrics Workshop 2009

The `actuar` package provides some functionalities for actuarial science, which is just "risk management" with a different vocabulary:

- Distributions often used to model losses (on top of the usual dpqr functions, you also have moments $E[X^k]$, limited moments $E[\text{Min}(X,x)^k]$ and moment generating functions $E[e^{tX}]$);
- *Minimum distance estimators* (MDE) for the Cramer-von Mises, modified χ^2 or layer average severity distances (these are not clearly defined);
- Handling (plotting, fitting) of censored data (both sides);
- Various estimators of the distribution of $X_1 + \cdots + X_N$ from those of N and the X_i (assumed iid): Panjer algorithm (uses discretized distributions), simulations, normal approximations (these algorithms are not detailed);
- Ruin models, *i.e.*, estimation of

$$P\big[\exists t > 0 \ : \ U(t) < 0\big]$$

where

$$\begin{aligned} U(t) &: \text{surplus at time } t \\ u &: \text{initial surplus} \\ c_k &: \text{premium collected at time } k \\ N(t) &: \text{number of claims in } [0,t] \\ X_n &: \text{value of the } n\text{th claim} \end{aligned}$$

$$U(t) = u + \sum_{k \leqslant t} c_k - \sum_{n \leqslant N(t)} X_n$$

from the distributions of N and X;
- Simulations of hierarchical models.

An overview of random number generation
C. Dutang
RMetrics Workshop 2009

The default random number generator (RNG) in R (the `runif` function) is the **Mersenne twister** (MT), a linear congruential RNG modified by some bitwise operations. The `randomtoolbox` package implements other RNG (mainly generalized MT: WELL, SFMT) and quasi-random number generators (aka **low discepancy sequences**):

- For the **Van der Corput** sequence, choose a prime number p, write successive numbers $1, 2, 3, \ldots, n$ in base p, $n = \sum_j a_j p^j$ and return $\phi_p(n) = \sum_j a_j / p^{j+1}$; if you choose several prime numbers, you have a multidimensional low discrepancy sequence, the **Halton sequence**; in R, the `halton(n,k)` function returns the first n elements of the k-dimensional Halton sequence;
- The **torus algorithm** (or **Kronecker sequence**) is

$$u_n = (\text{frac}\, n\sqrt{p_1}, \cdots, \text{frac}\, n\sqrt{p_d})$$

where $\text{frac}\, x = x - \lfloor x \rfloor$ is the fractional part.

RNGs can be tested by looking at

- The histogram of generated data;
- The autocorrelation function (ACF);
- The order test, which checks in which order x_{i-1}, x_i and x_{i+1} are);
- The Poker test: take d consecutive numbers from your sequence, count how many there are in each $[(k-1)/d, k/d[$ interval ($k \in [\![1, d]\!]$), compare with the expected values using a χ^2 test.

Discrepancy can be defined as

$$D_n(x, A) = \frac{1}{n} \#\big\{ \, i \in [\![1, n]\!] \ : \ i \in A \, \big\}$$

$$D_n^\infty(x, \mathscr{P}) = \sum_{A \in \mathscr{P}} |D_n(x, A)|$$

for some $\mathscr{P} \subset \mathscr{P}([0,1]^d)$.

GMM and GEL with R
P. Chaussé
RMetrics Workshop 2009

The *generalized method of moments* (GMM) fits a model to your data by considering several quantities or *moments* (e.g., average, variance, median, quantiles, higher moments, truncated moments, etc.) and fine-tuning the parameters so that the theoretical quantities be as close from the observed ones as possible. More precisely, find several moments $g = (g_1, \dots, g_n)$ so that $E[g(\theta, X)] = 0$ and compute

$$\hat{\theta} = \operatorname*{Argmin}_{\theta} \|\bar{g}(\theta)\|^2$$
$$\bar{g}(\theta) = \frac{1}{T}\sum_t g(\theta, x_t).$$

Since the moments are not "independent", the Euclidian distance is not the best choice: you will prefer

$$\hat{\theta} = \operatorname*{Argmin}_{\theta} \bar{g}(\theta)' W \bar{g}(\theta)$$

for some W. The most efficient estimator is obtained with

$$W = \sum_{s\in\mathrm{Z}} \mathrm{Cov}(g(\theta, x_t), g(\theta, x_{t+s})$$

which can be estimated (with an HAC (heteroscedasticity and autocorrelation consistent) estimator) as

$$\hat{\Omega}(\theta^*) = \sum_{|s|\leqslant T-1} k_h(s)\hat{\Gamma}_s(\theta^*)$$
$$\hat{\Gamma}_s(\theta^*) = \frac{1}{T} g(\theta^*, x_t) g(\theta^*, x_{t+s})'$$

k_h : some kernel

For instance, one could try any of the following:

- Estimate θ^* with $W = 1$; compute $\hat{\Omega}(\theta^*)$; estimate $\hat{\theta}$;
- Iterate the above until convergence;
- Optimize properly:

$$\hat{\theta} = \operatorname*{Argmin}_{\theta} \bar{g}(\theta)' \hat{\Omega}(\theta) \bar{g}(\theta).$$

The limiting distribution is known.

In finance or economy, you can use the CAPM, the utility function and other meaningful quantities to define moments.

In R, just provide your moments and your data to the `gmm` function.

A tale of two theories: Reconciling random matrix theory and shrinkage estimation as methods for covariance matrix estimation
B. Rowe
RMetrics Workshop 2009

Both methods try to remove the noise in the spectrum of the sample covariance matrix: random matrix theory (RMT) violently shrinks the low values to zero; shrinkage smoothly shrinks all values to their mean (or to a predefined value). Combining the two methods turns out to be a bad idea.

Portfolio optimization in R
G. Yollin
R in Finance 2009

Most portfolio optimization problems can be solved within R:

- For mean-variance optimization with no constraints (except perhaps a long-only one), use the `portfolio.optim` function in the `tseries` package;
- For the whole efficient frontier, call the function inside a loop;
- For the maximum Sharpe ratio portfolio, search on that efficient frontier;
- For mean-variance optimization with linear constraints (linear transaction costs, maximum weights, sectors or countries, etc.), use `solve.QP` in the `quadprog` package;
- Expected shortfall (ES, aka conditional value at risk, CVaR) optimization, is only a linear problem (but beware: it is a *scenario-based* optimization) is just a linear problem: you can use the `Rglpk_solve_LP` function in the `Rglpk` package (it can also do mixed integer linear programmin (MILP), but not quadratic programming);
- For more general constraints and/or performance/risk measures, such as

$$\Omega(r) = \frac{\int_r^\infty (1 - F(u))\, du}{\int_{-\infty}^r F(u)\, du}$$

  the drawdown (which can be plotted as an **underwater graph**, with the `chart.Drawdown` function), the **Rachev ratio** $R = \text{CVar}_\alpha^+ / \text{CVar}_\alpha^-$, you can check the `DEoptim` package.

A few remarks:

- Drawdown optimization looks very good in-sample, but it is a scenario-based optimization: how does it perform out-of-sample?
- There is no mention of constraints on the number of assets or of mixed integer quadratic programmins (MIQP): what about `Rsymphony` (it is open-source)?
- There is no mention of stochastic programming – but it is not reasonable (yet?) for more than 2 or 3 assets.

Working with xts and quantmod
J.A. Ryan
R in Finance 2009

The `xts` package relies on `zoo` to provide *time-based indexing*.

The `quantmod` package abstracts data access, from local (RData, CSV, SQLite, MySQL) or public sources (Google, Yahoo, Oanda, FRED). The `getSymbols` functions assigns to local variables of the same name.

There are also a few plotting functions (`chartSeries`, etc.).

Data can be loaded on-demand (lazily) and cached in memory or in a file.

For more, check the `TTR`, `blotter`, `PerformanceAnalytics` packages.

Statistics for Financial Engineering: Some R Examples
D. Ruppert
R in Finance 2009

After clearly explaning the difference between linear, non-linear and non-parametric regression, the author presents three applications of statistics in finance.

The **probability of default** (PD) is often modeled as

$$P[\text{default} \,|\, \text{rating}] = \exp(\beta_0 + \beta_1 \text{rating});$$

which can be transformed via a **Box-Cox transformation** $p \mapsto (\kappa + p)^\lambda$ (unless the problem dictates another family of transformations); you cannot naively take the logarithm, because the probability can be zero (many textbooks do this, and have to throw observations away, resulting in a biased estimator). Whatever transformation you choose, do not forget the **residual analysis**. (The presenter wrote a book about transformations.)

Most stylized facts about an interest rate time series $(R_t)_t$ are visible in the following plots:

$$R_t \sim t, \quad \Delta R_t \sim t, \quad \Delta R_t \sim R_{t-1} \quad (\Delta_t)^2 \sim R_{t-1}.$$

They can be modeled as a **diffusion process**

$$\Delta_t = \mu(R_{t-1}) + \sigma(R_{t-1})\varepsilon_t$$

where the drift μ and the volatility σ can be nonlinear (e.g., $\sigma(r) = \beta_0 r_1^\beta$) or **non-parametric**. With 50 years of data, non-parametric regression (e.g., with the `SemiPar` package – the presenter wrote a book on non-parametric estimation with the package author), non-parametric regression performs better: the parametric model will be outside the confidence interval of the non-parametric one. In this case, the residuals look GARCH

```
garch(x=res^2, order=c(1,1))
```

but the GARCH residuals still have some AR(1) noise

```
garchFit(
  formula = ~arma(1,0)+garch(1,1),
  data = res
)
```

and are far from Gaussian

```
garchFit(
  formula = ~arma(1,0)+garch(1,1),
  data = res,
  cond.dist = "std"
)
```

This 2-step process may be suboptimal.

The last example tries to predict future returns (starting with the worst):

- The average of the previous returns;
- The CAPM, forecast $= \beta \cdot$ previous returns;
- Bayesian shrinkage:

  $$\text{forecast} = \lambda\cdot\text{previous returns} + (1-\lambda)\cdot\text{market returns}$$

- The previous market returns (this is the best forecast, but all stocks end up with the same forecast).

The **bayesian shrinkage** can be implemented with WinBUGS/R2WinBUGS (OpenBugs/BRugs is no longer on CRAN, and OpenBugs was written in a obscure, commercial, non-textual, Pascal-like language (the source code is a binary file); JAGS is not mentionned):

$$\mu_i \sim N(\alpha, \sigma_1^2)$$
$$r_{i,t} \sim N(\mu_i, \sigma_2^2);$$

this is a data-driven shrinkage: the amount of shrinkage is σ_1^2/σ_2^2.

Risk Capital for Interacting Market and Credit Risk: VEC and GVAR models
K. Rheinberger
R in Finance 2009

International financial regulations (Basel II) assume that market and credit risk are independent and consider that assumption conservative: actually, the interaction between market and credit risk can be benign or malign. For small fluctuations, this can be seen from a Taylor expansion: let x be the risk factors (market, credit); let $f(x)$ be the value of your portfolio, then

$$f(x+\varepsilon) = f(x) + f'(x)\cdot\varepsilon + f''(x)\cdot\varepsilon\cdot\varepsilon + o(|\varepsilon|^2),$$

where the bilinear form $f''(x)$ can be positive, negative, or neither. This shows the problem for small fluctuations: we would like a decomposition

$$f(x+\varepsilon) = f(x) + g_{\text{market}}(x, \varepsilon_{\text{market}}) + g_{\text{credit}}(x, \varepsilon_{\text{credit}})$$

for large fluctuations as well – this works iff the portfolio is *separable*, *i.e.*, $f(x) = f_{\text{market}}(x) + f_{\text{credit}}(x)$.

ICA for multivariate financial time series
D. Matteson
R in Finance 2009

Since independent component analysis (ICA) is an optimization problem, you can easily add constraints; you could define a **constrained PCA** in the same way.

This presentation also tries to give technical details on how ICA is estimated (but it quickly becomes a list of formulas with no explanations) and how a time dimension can be added (but their "time-varying mixing matrix" looks like a moving window estimator).

Random portfolios: Practice and Theory
P. Burns
R in Finance 2009

Random portfolios can be seen as a kind of bootstrap or permutation test.

The basic idea is the following: since we add so many constraints, many things only depend on the constraints; random portfolios can measure the performance or risk coming from the constraints and isolate the value added by the portfolio manager.

Random portfolios (and matched portfolios) can replace a benchmark or a peer group: they will be less biased and lead to tests with a higher **power**; they can also incorporate more information, such as the position at the begining of the period and the turnover constraint.

Random portfolios can also help measure the (sometimes unwanted) impact of constraints: you can compare the performance and risk of random portfolios for several sets of constraints (e.g., loose or tight constraints on sector weights).

Generating random portfolios is problematic.

- A rejection strategy is not reasonable (there are too few portfolios satisfying the constraints);
- Algorithms to sample inside a polytope do not apply if there are non-linear constraints;
- You can compute an optimal portfolio for random returns, but this may be biased;
- You can use a random search: start with a random, unconstrained portfolio and minimize a penalty for broken constraints; however, you will be close to a binding constraint (this does not worry me: we are in high dimension, everything is close to the boundary) and the sampling will not be uniform (this is not very worrying: this is almost exactly what a portfolio manager does, except that he starts with non-random returns).

Matching portfolios
D. Kane
R in Finance 2009

To measure the performance of a portfolio, one may look at the average capitalization, the momentum, and the book market value of its constituents and compare with the corresponding quintile portfolio: but you have a single portfolio and the more factors you add, the emptier those quintile portfolios become. *Matching* views those portfolio characteristics (except those in which the portfolio manager claims expertise) as constraints and build a portfolio matching those constraints.

I do not see the difference with *random portfolios*.

Using R for hedge fund of funds management
E. Zivot
R in Finance 2009

When faced with non-gaussian data, use the Cornish-Fisher expansion to compute the value at risk (VaR) or the expected shortfall (ES, CVaR).

If the time series of returns do not have the same length: select the best risk factors; compute the exposure of the funds to those factors; simulate the missing returns; use those simulated returns to estimate the marginal contribution to risk (MCTR) or risk attribution for VaR or ES. This is available in the `PerformanceMetrics` package.

This reminds me of the problem of estimating a variance matrix $\mathrm{Var}(x_1, \dots, x_n)$ when there are many missing values: the maximum-likelihood estimator can be computed explicitely (this is "just" linear algebra, albeit horrible), but an MCMC simulation is easier to implementat (and less likely to be buggy). However, even for toy examples, the convergence is horribly slow.

The presentation mentions but does not tackle the following problems:

- Biases (survivorship, reporting, backfill);
- Serial correlation (performance smoothing; illiquid positions).

Performance Analysis in R
P. Carl and B. Peterson
R in Finance 2009

Clear presentation of what the `PerformanceAnalytics` package can do: measure, compare, decompose the performance or risk of portfolios.

Market Microstructure Tutorial
D. Rosensthal
R in Finance 2009

Market microstructure is the study of why markets are not efficient and how to profit from it (or, equivalently, help make them more efficient); this includes behavioural effects.

The *Glosten-Milgrom model* considers two traders: one (informed) always buys (or sells, this is fixed at the begining); the other (non-informed) buys or sells with probability 1/2 (they just buy or sell, the model does not consider the need to find a counterparty); only one trader (selected at random) trades at each point in time. We define the *bid* (respectively, *ask*) as the probability that the informed trader is buying (resp. selling) given that the last trade was a buy (resp. sell). The bid and ask converge towards the true price (0 or 1) (*i.e.*, the market goes where the informed trader wants: market manipulation is indistinguishable from price discovery).

The *Kyle model* considers a market maker facing informed and non-informed traders and sets the price to the expected true price given the total order size he sees; the informed trader chooses the order size to maximize profit. There again, information leaks very quickly.

Portfolio Optimization with R/RMetrics
D. Würtz
R in Finance 2009

Quick overview of what is available in RMetrics for portfolio optimization – there is an ebook with the same title (it excludes: quadratic (non-linear) constraints, integer constraints, non-linear objectives, Black-Litterman copula pooling (BLCP), quadratic lower partial moments (QLPM), copula tail risk: these will be covered in a second volume). The list of packages looks exhaustive: `Rglpk`, `Rsymphony`, `Rlpsolve`, `quadprog`, `Ripop`, `Rsocp`, `Rdonlp2`, `Rnlminb`.

Econometrics and practice: mind the gap!
Y. Chalabi and D. Würtz
R in Finance 2009

There is an unreasonable number of variants of the GARCH model, but fitting them poses many practical issues (bugs, parameter initialization, optimization schemes, etc.).

The presentation slides contain some sample code: it is surprisingly straightforward (it would be trickier to estimate a stochastic volatility model, *i.e.*, a model with two sources of randomness: you would need to consider a time series of (hidden) innovations).

The stationary assumption is often violated.

We might be tempted to use *robust* methods to throw outliers away, but we cannot afford it: we would underestimate risk. Instead, you can include them in the model, either as isolated points or as a different regime: **MS-GARCH** (Markow-Switching GARCH) is one such example.

Indirect inference can help fit a difficult-to-fit model g by using an (easier-to-fit) model f:

- Estimate the auxiliary model f in the data y: $\hat{\theta}$;
- For all possible values of the parameters ρ of the model g, generate a sample $\tilde{y}$, and estimate the auxilliary model f on it: $\tilde{\theta}(\rho)$.
- Find the parameters ρ that minimize the distance $\left\| \hat{\theta} - \tilde{\theta}(\rho) \right\|$.

Introduction to high-performance computing with R
D. Eddelbuettel
R in Finance 2009

You can profile your code with the `system.time`, `Rprof`, `Rprofmen`, `tracemem` functions; the `profr` and `proftools` package; the `prof2dot.pl`, `valgrind`, `kcachegrind`, `sprof`, `oprofile`, Google PerfTools tools.

Vectorization, with the `*apply` or `outer` functions, or just-in-time compilation with Ra (a set of patches for R, available for Debian/Ubuntu – a very search-engine-unfriendly name) and the `jit` package, can speed things up.

For the best performance, you can rewrite the most

time-consuming parts of the code in C/C++: with `.C`, the C function uses C types; with `.Call`, it uses R types (`SEXP`); with `Rcpp`, the conversion between R and C++ types (templates) is transparent. The `inline` package and the `cfunction` function allow you to put the C code (with `.Call` or `.C` conventions) directly inside the R code – as with Perl's `Inline::C` module.

In the other direction, `RInside` allows you to evaluation R code from a C/C++ program.

Other high-performace topics were not mentionned:

- Parallel computing: MPI, nws, snow;
- Out-of-memory computations: `biglm`, `ff`, `bigmatrix`;
- Scripting and automation.

A latent-variable approach to validate credit rating systems with R
B. Grün et al.
R in Finance 2009

Credit rating of several firms (obligors) by several raters (banks, rating agencies) can be modeled as a mixed model: let the observed score $S_{i,j}$ be the probit of the announced probability of default for obligor i according to rater j:

$$S_{i,j} = S_i + \mu_j + \sigma_j Z_{i,j},$$

where S_i is the latent (consensus) rating and μ_j the bias of rater j.

This can be generalized into a dynamic model by assuming the latent probability is AR(1) and examined, by Gibbs sampling, with the `rjags` and `coda` packages. In particular, you can look at: bias and variance of raters, consensus rating; and the residuals can be further analyzed.

In practice, you have to map the ordinal rating of the rating agencies (AAA, etc.) into probabilities: this hides the bias.

Quantile regression in R
R. Koenker
R in Finance 2009

The definition of a quantile can be formulated as a linear optimization problem and this generalizes readily to a regression setup

$$\alpha_t = \operatorname*{Argmin}_{\alpha\in\mathrm{R}} \sum_{i=0}^{n} \rho_\tau(y_i - \alpha)$$
$$\rho_\tau(u) = (\tau - 1_{u<0})u.$$

Even with a linear model, quantile regression can highlight non-linear features, such as the conditional distribution of the data (this is not as general as the distribution of $y|x$, but almost: there is a model, which assumes a linear relation between the quantiles of x and (a linear combination of) y, but that relation can depend on the quantile; in particular, we can end up with non-gaussian, and even multimodal, distributions) or the dependence of the regression coeffients on the quantile (for an additive gaussian model, we would expect the intercept to be the inverse cumulative distribution function of the quantile and the other coefficients to be constant).

In a time series setup, you can use quantile regression to estimate a **quantile autoregression** (QAR): it it will be non-gaussian, but still partly parametric.

Quantile regression is actually equivalent to CVaR (conditional value at risk) portfolio optimization; by approximating the weight function of a spectral risk measure by a locally constant function, quantile regression can be used to optimize portfolios for any "pessimistic" risk measure (this has nothing to do with quantile regression: it applies to linear programming in general).

The presentation ends up with a non-gaussian regression

$$\text{score} \sim \text{team 1} + \text{team 2} + \text{home advantage}$$

(all the predictors are qualitative variables, the variable to predict is a count variable: this would classically be estimated by a Poisson model – but we would have to check this distributional assumption). With quantile regression, we keep the linearity of the model, but remove any distributional assumption; the model can be estimated as a (not unreasonably) large but sparse linear program and one can *sample* from the distribution of scores, in a half-bayesian fashion.

Quantile regression can be seen as a prior-less bayesian regression with no distributional assumptions.

Copula-based non-linear quantile autoregression
X. Chen et al.

A copula-based Markov model (for a stationary time series $(Y_t)_t$) just specifies the copula C of (Y_{t-1}, Y_t) and the marginal distribution of Y_t. This is often estimated by a 2-step precedure: first, estimate the marginal distribution, then, estimate the copula (using either the empirical marginal distribution or the fitted one).

One can look at the τth quantile of $Y_t|Y_{t-1} = x$:

$$F^{-1}\big(C(\tau,\cdot)^{-1}(F(x))\big).$$

The article considers the case of a parametric copula and marginal distribution, but those parameters depend on the quantile τ This is more robust to model misspecification.

Quantile autoregression
R. Koenker and Z. Xiao (2006)

Quantile autoregression (QAR) is a special case of **random coefficient autoregression** (RCAR)

$$y_t = \theta_0(U_t) + \theta_1(U_t)y_{t-1} + \cdots + \theta_p(U_t)y_{t-p}$$
$$U_t \sim U(0,1).$$

For instance

$$\theta_0(\tau) = \sigma\Phi^{-1}(\tau)$$
$$\theta_1(\tau) = \text{Min}(\gamma_0 + \gamma_1\tau, 1)$$

leads to a mixture of unit root or explosive behaviour (when $\theta_1 = 1$) and mean-reverting behaviour (when $\theta_1 \in (0, 1)$) – this is actually sufficient to make the process stationary.

Since the functions θ are unknown, this is a non-parametric model. (Most of the theoretical troubles come from the requirement that those functions be monotonic.) You can test for *asymmetric dynamics* by checking if θ_1 depends on τ at all.

March madness, quantile regression bracketology and the Hayek hypothesis
R. Koenker and G.W. Bassett

More details on this example.

Issues on quantile autoregression
J. Fan and Y. Fan

Quantile autoregression (QAR) is related to **functional coefficient autoregression** (FCAR) models

$$y_t = \alpha_0(U_t) + \alpha_1(U_t)Y_{t-1} + \cdots + \alpha_p(U_t)Y_{t-p} + \varepsilon_t$$

where U_t is observed. The special case $U_t = Y_{t-d}$ is known as **functional autoregression** (FAR). Identifiability is a real problem, and in case of lack of monotonicity, the estimators are not consistent.

Monitoring networked applications with incremental quantile estimation
J.M. Chambers et al.
`arXiv:0708.0302`

Sequential quantile estimation algorithms compute quantiles in a "cumulative" (online) way, as data arrives:

- Exact algorithms exist (e.g., based on the quick sort) to track a single quantile, but they have to keep a potentially large buffer of data; they can be modified to track a set of quantiles;
- Stochastic approximation is similar, but assumes the data has a continuous distribution to approximate between the quantiles.

Incremental quantile (IQ) estimation is a quick-and-dirty algorithm that allows both online quantile estimation and *aggregation* of quantiles from distributed agents; it uses cumulative distribution functions (CDF), which are easy to aggregate, rather than quantiles:

- On each agent, wait until you have enough data; turn it into a cumulative distribution function (CDF); blend it with the previous CDF estimate;
- From time to time, send it to the server, and start the computation afresh;
- On the server, aggregate the CDFs received from the agents and compute the desired quantiles.

The CDFs can be stored and sent as a set of quantiles (for increased precision, we may want more quantiles than the final users actually receive).

This can be useful in distributed QoS (quality of service) measurement.

Event study: a change-point model approach
J.H. Yoon
R in Finance 2009

Hidden variable (aka latent variable, or regime switching if that variable is discrete) models can be estimated via Monte Carlo simulations. In this example, we consider spells of iid gaussian random variables and estimate the model by Gibbs sampling, with a bayesian prior:

- Estimate the state at each point in time;
- Estimate the parameters in each spell;
- Estimate the transition probabilities;
- Iterate.

(At each step, we consider the whole time series.) This can be repeated with a different number of regimes, and the best model can be selected by looking at the *Bayes factor*.

In a financial setting, breakpoints need not coincide with announcements: they can occur before.

Detecting structural breaks in tail behavior
W.H. Liu
R in Finance 2009

The *Hills estimator* of the tail index (and the tail index itself) only makes sense for power law tails and, even in that case, is known to be very noisy. Replacing the power law by a general extreme value (GEV) distribution does not solve those problems: tests for structural breaks (I do not know them: SupLN, OLS-based CUSUM, Nyblom-Hansen, generalized M-fluctuation) return a surprisingly high number fo potential regime changes.

For risk management purposes, estimators of GEV models are as problematic as the Hills estimator.

strucchange: an R package for testing for structural change in linear regression models
A. Zeilis et al.

The `strucchange` package implements tests based on the following **empirical fluctuation processes**:

- OLS-CUSUM: cumulated sum of the residuals (a brownian bridge);
- recursive CUSUM: cumulated sums of the residuals, where the ith residual comes from the model estimated on the first i observations;
- Recursive MOSUM: idem, on a moving window;

– Fluctuations: parameters of the model estimated on an expanding window, suitably renormalized (contrary to the residuals, this is a multidimensional process);
– Moving estimate: idem, on a moving window.

In case of a structural change, these processes "fluctuate more"; they can be compared with boundaries of the form $b(t) = \lambda$ (MOSUM: it is stationary), $b(t) = \lambda\sqrt{t}$ (brownian motion) or $b(t) = \lambda\sqrt{t(1-t)}$. However, the crossing probability is easier to compute for linear boundaries: we lose some power at the begining and end of the interval.

```
r <- epf(y ~ x, type="fluctuation")
plot(r)
sctest(r)
```

Instead of those *significance tests*, which have no clear alternative hypothesis, you may prefer an F test, which tests against the alternative that there are two different models, one before and one after an (unknown) breakpoint; they compare the sum of squared residuals for the model with and without a breakpoint at observation τ with an F statistic F_τ and aggregate them as $\sup F_\tau$, $\text{Mean}\, F_\tau$ or $\log \text{Mean} \exp F_\tau$.

```
f <- Fstats(y~x)
sctest(f)
```

The package also provides **monitoring**.

A unified approach to structural change tests based on F statistics, OLS residuals and ML scores
A. Zeileis

The following classes of structural change tests are actually generalized M-fluctuation tests, built from the empirical fluctuation process (efp):

– CUSUM (the absolute maximum of the cumulative sums of the residuals rescaled by an estimate $\hat{\sigma}^2$ of the error variance – it is not a real test, there is no clear alternative hypothesis, but it is often used as an exploratory tool);
– $\text{SupLM} = \sup_{t\in[0,1]} \frac{\|\text{efp}_t\|_2^2}{t(1-t)}$ (in the alternative, the model parameters have two values, before and after some (unknown) breakpoint);
– The Nyblom-Hansen statistic, where in the alternative H_1, the parameters follow a random walk, boils down to
$$\frac{1}{n}\sum_{i=1}^{n} \left\|\text{efp}_{i/n}\right\|_2^2.$$

To detect multiple breakpoints, you may want to use a moving sum (MOSUM) instead of a cumulative sum.

These tests can be extended to a **monitoring** setup, where the observations arrive one by one: estimate the model on $[0,1]$ and compute the empirical fluctuation process beyond 1; the critical values become a curve $b(t)_{t>1}$ instead of a constant.

Permutation tests for structural change
A. Zeileis and. T. Hothorn

The SupF (or SupLM: the test statistic can be transformed to look like an F or a Student T one) tests whether the average of a series of random variables $(X_{k/n})_{k\in[\![1,n]\!]}$ is a constant μ against the alternative hypothesis H_π that it is μ_1 before some breakpoint $\tau \in [0,1]$ and μ_2 after, by considering the statistic
$$\frac{\text{RSS}_\pi}{\text{RSS}_0},$$
suitably renormalized
$$Z_\pi = \sqrt{(n-1)\left(1 - \frac{\text{RSS}_\pi}{\text{RSS}_0}\right)},$$
where RSS_0 and RSS_π are the sums of squared residuals under H_0 and H_π, and aggregating them as
$$D = \max_\pi Z_\pi.$$
This statistic can be compared with the following distributions:

– $\mathscr{D}_\infty$: the *unconditional limiting distribution*, *i.e.*, the limiting distribution, for $n \to \infty$; it is
$$\sup_{\pi\in[0,1]} \pi(1-\pi)B(\pi),$$
where B is a brownian bridge;
– $\mathscr{D}_{\sigma|Y}$: the *conditional distribution*, obtained by randomly permuting the observations (usually computed via simulations; only in very rare cases is it analytically tractable);
– $\mathscr{D}_{\infty|Y}$, the *asymptotical conditional distribution*, which can be computed analytically:
$$Z \sim N(0,\Sigma)$$
$$Z_{\pi,\tau} = \frac{\pi(1-\tau)}{\sqrt{\pi(1-\pi)\tau(1-\tau)}} \qquad (\pi \leqslant \tau)$$

This can be generalized to structural changes in binary variables, multivariate series, stratified observations, parametric models.

Implementing a class of structural change tests: an econometric computing approach
A. Zeileis

Structural change tests can be performed as follows:

– Compute the residuals of your model, $(r_i)_{i\in[\![1,n]\!]}$;
– Robustify those residuals, $\big(\psi(r_i)\big)_{i\in[\![1,n]\!]}$, where ψ is an M-estimation scale function; we only want the sum of the robustified residuals to be zero;
– Compute their cumulative sum process $(Z_t)_{t\in[0,1]}$ (the article is not very clear if we take the cumulative sum of the residuals or build a process from the model coefficients estimated on an expanding window: we cannot get a multidimensional process with just the residuals);

– Estimate the asymptotic covariance matrix of the scores $\hat{J}$; we then have

$$\mathrm{Cov}(Z_t, Z_s) = \mathrm{Min}(t, s)\hat{J};$$

you can (should) use an HAC (heteroskedasticity and autocorrelation consistent) estimator for J;
– Compute the decorrelated fluctuation process

$$\mathrm{efp}_t = \hat{J}^{-1/2} Z_t;$$

it converges to a brownian bridge (since the residuals sum up to zero, we have $Z_0 = Z_1 = 0$); you may choose to use the whole matrix $\hat{J}$ or just its diagonal elements;
– Aggregate this multidimensional process, for instance:
  · Take the maximum of the components at each point in time, then take the maximum accross time (maxBB);
  · Take the L^2 norm of the components at each point in time; then take the average over time (Cramer von Mises statistic);
  · Take the range (*i.e.*, $\mathrm{Max} - \mathrm{Min}$) over time, for each component; then take the maximum of those ranges (range test).

This is implemented in the `strucchange` package and will work for any model for which you can compute the residuals (Poisson, logistic, beta regressions, etc.).