# *Mathematica* commands summary (cheat sheet)

Hugo Touchette
National Institute for Theoretical Physics (NITheP), Stellenbosch, South Africa
Last updated: October 15, 2013
See last page for *Mathematica* 8 updates

**Abbreviations**

| | |
|---|---|
| approx | Approximation |
| arg | Argument |
| char | Character |
| cnt | Counter |
| cond | Condition |
| dist | Distribution |
| elem | Element |
| eqn | Equation |
| eval | Evaluation |
| expr | Expression |
| fct | Function |
| img | Image |
| patt | Pattern |
| rand | Random |
| str | String |
| val | Value |
| var | Variable |

## Equalities, assignments [DC]

| | | |
|---|---|---|
| `=` | Assign value | |
| `:=` | Delayed eval | Also used for repeated eval |
| `==` | Equality | |
| `=.` | Clear value | Same as `Clear` |
| `===` | Boolean test | Same as `SameQ` |
| `f[x_]:=f[x]=...` | Remember computed vals | |

## Rules and replacements [DC, Tut]

| | | |
|---|---|---|
| `patt->expr` | Transformation rule | Can be assigned to names |
| `patt:->expr` | Delayed rule | Same as `:>` |
| `expr/.rule` | Replacement | Same as `ReplaceAll` |
| | `1+2 x/.x->3` | Output: 7 |
| `expr/.{rule1,rule2}` | Multiple replacements | |
| `expr//.rule` | Replace until no changes | Same as `ReplaceRepeated` |

## Applications, functions and maps [DC, Tut]

| | | |
|---|---|---|
| `f@expr` | Prefix form `f[expr]` | |
| `expr//f` | Postfix form `f[expr]` | |
| `x~f~y` | Infix form `f[x,y]` | |
| `f@@expr` | Replaces the head of `expr` by `f` | Same as `Apply[f,expr]` |
| | `Plus@@{1,2}` | Output: $1+2=3$ |
| | Similar to: `f[x]/.f->g` | Output: `g[x]` |
| `f/@list` | Map `f` on `list` | Same as `Map[f,list]` |
| `f/@expr` | Map `f` on parts of `expr` | |
| | `Map[f,a+b]` | Output: `f[a]+f[b]` |
| `Function[x,body]` | Function with arg `x` | Same as pure functions |
| | `Function[x,x^2][n]` | Output: $n^2$ |
| | `Map[Function[x,x^2],a+b]` | Output: $a^2+b^2$ |
| `Function[{vars},...]` | Many args | |
| `body&` | Pure (anonymous) function | |
| | `(#^2&)[5]` | Output: 25 |
| | `Map[#^2&,a+b]` | Output: $a^2+b^2$ |
| | `(1+g[#])&/@{1,2}` | Output: `{1+g[1],1+g[2]}` |
| `#n` | $n$th arg in pure function | |
| `##` | Seq of all args | |

## Formatting [DC]

| | |
|---|---|
| `FullForm` | Display full head form |
| `MatrixForm` | Display in matrix form |
| `Column` | Display in column form |
| `TableForm` | Display in tabular form |
| `TreeForm` | Display in tree structure |

## Shortcuts [DC]

| | |
|---|---|
| Ctrl-@ | $\sqrt{\cdot}$ |
| Ctrl-/ | Fraction |
| Ctrl-^ | Superscript |
| Ctrl-_ | Subscript |
| Ctrl-Enter | New row |
| Ctrl-, | New column |

## Patterns [DC, Tut]

| | | |
|---|---|---|
| `_` | Blank: any expr | |
| `__` | Double blank: any seq of expr | |
| `___` | Triple blank: zero or more args | |
| `x_` | Any expr named `x` | |
| `x_h` | Expr named `x` with head `h` | |
| `x` | Exact matching `x` | |
| | `x+2/.x->2` | Output: 4 |
| `x:patt` | Named expr matching `patt` | |
| | `f[a^b]/.f[x:_^_]->p[x]` | Output: `p[a^b]` |
| `x_:v` | Optional arg with default val `v` | |
| `x_h:v` | Typed arg with optional val `v` | |
| `f[n_]` | `f` with any named arg `n` | |
| `f[n_,n_]` | `f` with identical args | |
| `x^n_` | `x` to any named power `n` | |
| `x_^n_` | Any expr to any power | |
| `a_+b_` | Sum of two named exprs | |
| `{a1_,a2_}` | List of two named exprs | |
| `patt/;cond` | Pattern with cond | Read: `patt` such that `cond` |
| | `(x_/;NumberQ[x]&&Im[x]==0)` | Any real number |
| `rule/;cond` | Rule with cond | |
| `patt?test` | Pattern matching boolean test | |
| | `f[x_?NumberQ]:=x+2` | |
| `(patt1\|patt2\|...)` | `patt1` or `patt2` or ... | |
| | `{1,x,x^2,y^3}/.(x\|x^n_)->q` | Output: `{1,q,q,y^3}` |
| `expr/.rule` | Replacement in `expr` with `rule` | |
| | `x+2/.x->2` | Output: 4 |
| | `f[a]+f[b]/.f[x_]->x^2` | Output: `a^2+b^2` |
| | `Position[f[a],g[b],f[b],f[x_]]` | Output: $\{\{1\},\{2\}\}$ |
| | `{1,x,x^2,x^3}/.x^n_->r[n]` | Output: `{1,x,r[2],r[3]}` |
| `Except[c]` | Anything except $c$ | |
| `Except[c,patt]` | Pattern `patt` except $c$ | |
| | `Cases[{1,0,2},Except[0]]` | Output: `{1,2}` |
| | `Cases[{1,x,0},Except[0,_Integer]]` | Output: `{1}` |

## Type specification

| | | |
|---|---|---|
| `Integer` | $\mathbb{N}$ | Esc-dsN-Esc |
| `Real` | $\mathbb{R}$ | Esc-dsR-Esc |
| `Complex` | $\mathbb{C}$ | Esc-dsC-Esc |
| `Rational` | $\mathbb{Z}$ | Esc-dsZ-Esc |
| `List` | List type or head | |
| `Symbol` | Any symbol | |
| `String` | String type of head | |
| `_Head` | Type `Head` | |
| | `MatchQ[x^2,_Power]` | |

## Files [DC]

| | |
|---|---|
| `Import["file"]` | Import file or url |
| `Export["file",expr]` | Export `expr` in file |
| `ImageResolution->pts` | Image resolution in pts |
| `ReadList["file"]` | Returns list from file |
| `<<File‘` | Load content of file |

## Lists [DC, Tut]

| | | |
|---|---|---|
| `{i,imax}` | Simple counter from 1 to `imax` | |
| `{i,imin,imax,di}` | Counter with step `di` | |
| `Table[expr,cnt]` | Generate list | Multiple counters possible |
| `list[[i]]` | $i$th elem (list or expr) | Same as `Part[list,i]` or `list[[i]]` |
| `list[[i,j]]` | $(i,j)$ elem | Same as `list[[i]][[j]]` |
| `list[[i;;j]]` | Range $i$ to $j$ | Same as `Part[list,i;;j]` |
| `Drop[list,n]` | First $n$ elems dropped | Last if `-n` |
| `Drop[list,{n}]` | $n$th elem dropped | |
| `Extract[expr,list]` | Extract in `expr` at pos `list` | |
| `Array[f,n]` | $\{f(1), f(2), \ldots, f(n)\}$ | |
| | `Array[1+#^2&,2]` | Output: $\{2, 5\}$ |
| `Range[n]` | $\{1, 2, 3, \ldots, n\}$ | Same as `Array[#&,n]` |
| `Range[nmin,nmax,dn]` | Full syntax | |
| `Position[expr,patt]` | List matching positions | |
| `Count[list,patt]` | Number of matches | |
| `Min[list], Max[list]` | Min, max of list | Same as: `Min[i,j,...]` |
| `Append[expr,elem]` | `expr` with `elem` added at end | |
| `AppendTo[expr,elem]` | Append and update `expr` | Same as `expr=Append[expr,elem]` |
| `Prepend[expr,elem]` | Inset at beginning | See also: `PrependTo` |
| `Insert[list,elem,n]` | Insert `elem` in `list` at pos $n$ | |
| `Delete[expr,elem,n]` | Delete `elem` in `list` at pos $n$ | |
| `ReplacePart[expr,i->new]` | Replaces `expr[[i]]` by `new` | |
| `ReplaceList[expr,rule]` | Replace according to `rule` | |
| `Flatten[list]` | Flattens nested lists | |
| `Flatten[list,n]` | Flattens to level $n$ | |
| `Take[list,n]` | Take first $n$ elem | |
| | `Take[#,1]&/@{{1,2},{3,1}}` | Output: $\{\{1\},\{3\}\}$ |
| `Cases[list,patt]` | Elems of `list` matching `patt` | |
| | `Cases[{2,x^2},x^n_]` | Output: $x^2$ |
| `Cases[list,patt,n]` | First $n$ matching `patt` | |
| `Cases[list,rule]` | Results for which `rule` applies | |
| `Select[list,boolean]` | Boolean selection in `list` | |
| `Position[expr,patt]` | Pos where `patt` appears | |
| `Join[l1,l2,...]` | Join lists | |
| `Union[l1,l2,...]` | Set union | Same as $\cup$ |
| `Riffle[{e1,e2,...},x]` | `{e1,x,e2,x,...}` | |
| `Intersection[l1,l2,...]` | Set intersection | Same as $\cap$ |
| `Subsets[list]` | All subsets of `list` | |
| `DeleteDuplicates` | Delete duplicates from list | |
| `Thread[f[{a,b,c}]]` | `{f[a],f[b],f[c]}` | |
| `Sort[list]` | Sorts `list` | |
| `Reverse[list]` | `list` in reverse order | |
| `PadLeft[list,n,x]` | Add `x` $n$ times left of `list` | Pad with 0 if no `x` |
| `PadRight[list,n,x]` | Add `x` $n$ times right of `list` | |
| `Partition[list,n]` | Partition `list` in $n$ sublists | |
| `Split[list]` | Splits `list` in identical elems | |
| `Tuples[list,n]` | $n$-tuples of `list` | |
| `Permutations[list]` | Permutations | |
| `Outer[f,l1,l2]` | Outer (cartesian) product | |

## Boolean functions [DC]

| | |
|---|---|
| `SameQ[x,y]` | Same? (`x===y`) |
| `OrderedQ[list]` | `list` ordered? |
| `IntegerQ[x]` | $x$ integer? |
| `MemberQ[list,patt]` | Elems match `patt`? |
| `MatchQ[expr,patt]` | `expr` matches `patt`? |
| `ValueQ[expr]` | `expr` has value? |
| `AtomQ[expr]` | `expr` is atomic? |
| `FreeQ[expr,patt]` | `expr` free of `patt`? |
| `EvenQ[n]` | $n$ even integer? |
| `OddQ[n]` | $n$ odd integer? |
| `PrimeQ[n]` | $n$ prime integer? |
| `NumberQ[expr]` | `expr` a number? |
| `NumericQ[expr]` | `expr` numerical? |
| `PolynomialQ[expr,var]` | `expr` a polynomial? |
| `VectorQ[expr]` | `expr` 1D array (list)? |
| `MatrixQ[expr]` | `expr` 2D array (list)? |
| `DigitQ[str]` | All digits in `str`? |
| `LetterQ[str]` | All letters in `str`? |

## Programming [DC, Tut1, Tut2]

| | |
|---|---|
| `Do[expr,cnt]` | Do loop |
| `While[test,body]` | While loop |
| `If[cond,t,f]` | If statement |
| `Return[expr]` | Output `expr` and stop |
| `Module[{x,...},expr]` | Proc with local vars (vars localization) |
| `Block[{x,...},expr]` | Eval with global vars |
| `With[{x=x0,...},expr]` | Local constants |
| `Trace[expr]` | Trace eval of `expr` |
| `Compile[{vars},expr]` | Compiled expr |

| | |
|---|---|
| `i++` | Incrementation |
| `i--` | Decrementation |
| `i+=di` | Increment by `di` |
| `i-=di` | Decrement by `di` |
| `i*=c` | Multiply by $c$ and override val |
| `i/=c` | Divide by $c$ and override val |

## Strings [DC, Tut]

| | | |
|---|---|---|
| `"chars"` | String | |
| `ToString[expr]` | Transforms `expr` to string | |
| `StringJoin[str1,str2,...]` | Join strings | Same as <> |
| `StringLength[str]` | Number of chars in `str` | |
| `StringTake[str,n]` | $n$ first chars of `str` | Last if `-n` |
| `StringDrop[str,n]` | First $n$ chars dropped | Last if `-n` |
| `StringInsert[str1,str2,n]` | Insert `str2` at pos $n$ | |
| `StringPosition[str,substr]` | Start and end of `substr` in `str` | |
| `StringCount[str,substr]` | Number of occurrences of `substr` | |
| `StringReplace[str,rule]` | Replace with rule | |
| `StringMatchQ[str,patt]` | `patt` appears in `str`? | |
| `StringFreeQ[str,patt]` | `str` free of `patt`? | |
| `StringSplit[str]` | Split where whitespace | |
| `StringSplit[str,patt]` | Split according to `patt` | |
| `Sort` | Alphabetical (canonical) sort | |
| `DateString["elem"]` | String version of date | |
| `Characters[str]` | List of chars | |
| `ToUpperCase[str]` | Capitalize | |
| `ToLowerCase[str]` | Lower case | |
| `CharacterRange["char1","char2"]` | List from `char1` to `char2` | |
| `patt..` | `patt` followed by symbols | |
| `Whitespace` | White space | |

## Dynamic objects [ DC, Tut2 ]

| | |
|---|---|
| `Dynamic[expr]` | Dynamically updated expr |
| `DynamicModule[{vars},...]` | Module with dynamical vars |
| `Manipulate[expr,{u,...}]` | Interactive widget |

## Useful functions

| | |
|---|---|
| `Sum[f,{i,imin,imax,di}]` | Sum |
| `Product[f,{i,imin,imax,di}]` | Product |
| `Mod[m,n]` | Remainder of $m/n$ |
| `Solve[f[x]==0,x]` | Solves polynomial eqn |
| `Expand[p]` | Expand polynomial `p` |
| `Factor[p]` | Factorize `p` |
| `Length[expr]` | Number of elems in `expr` |
| `Numerator[expr]` | Numerator of `expr` |
| `Denominator[expr]` | Denominator of `expr` |
| `Options[cmd]` | Display options for `cmd` |
| `SetOptions[cmd]` | Set options for `cmd` |
| `Evaluate[expr]` | Explicit eval |
| `Piecewise[{{val,test},...}]` | Piecewise definition |
| `InverseFunction[f][x]` | Inverse fct |
| `Nest[f,x,n]` | $n$-fold fct composition |
| `NestList[f,x,n]` | List of compositions |
| `FixedPoint[f,x]` | Iterate until fixed point |
| `FixedPointList[f,x]` | Iteration list |
| `NestWhile[f,x,test]` | While iteration |
| `NestWhileList` | While iteration list |
| `LinearModelFit` | Linear model |
| `NonlinearModelFit` | Nonlinear model |
| `GraphPlot[m]` | Graph (network) for adjacency $m$ |
| `GraphData["name"]` | Graph with `name` |
| `FinancialData["tag","date"]` | Financial data from `date` |
| `CountryData["tag","prop"]` | Country data |
| `WeatherData["tag","prop"]` | Weather data |

## Probability and statistics [DC]

| | | |
|---|---|---|
| `RandomReal[{a,b}]` | Uniform rand real in $[a,b]$ | From $[0,1]$ if no arg |
| `RandomReal[dist]` | Rand real from `dist` | |
| `RandomInteger[{i,j}]` | Uniform integer in $[i,j]$ | Uniform 0 or 1 if no arg |
| `RandomInteger[dist]` | Rand integer from `dist` | |
| `RandomSample[list,n]` | $n$ samples from `list` | |
| `RandomSeed[]` | Reset seed | |
| `PDF[dist,x]` | Density fct of `dist` | |
| `CDF[dist,x]` | Cumulant density fct of `dist` | |
| `Histogram[list,w]` | Histogram (bin width $w$) | Option:`"Probability"` `"ProbabilityDensity"` |
| `BinCounts[list,w]` | Counting dist of `list` (bin width $w$) | |
| `Mean[list]` | Also `Mean[dist]` | |
| `Variance[list]` | | |
| `StandardDeviation[list]` | | |
| `ChiSquareDistribution[`$\nu$`]` | | |
| `ExponentialDistribution[`$\lambda$`]` | | |
| `NormalDistribution[`$\mu$`,`$\sigma$`]` | | |
| `BinomialDIstribution[n,p]` | | |
| `BernoulliDistribution[p]` | | |

## Linear algebra [DC]

| | |
|---|---|
| `a.b` | Dot or matrix product |
| `Cross[a,b]` | Cross product |
| `Norm[a]` | Euclidean norm |
| `IdentityMatrix[n]` | $n \times n$ identity matrix |
| `Diagonal[m]` | Diagonal of $m$ |
| `Diagonal[m,k]` | $k$th elem in diagonal of $m$ |
| `Dimensions[m]` | Dimensions of $m$ |
| `Inverse[m]` | Inverse square matrix $m$ |
| `Det[m]` | Determinant square matrix |
| `Tr[m]` | Trace matrix or tensor $m$ |
| `Transpose[l]` | Transpose first two levels |
| `Eigenvalues[m]` | Eigenvals square matrix |
| `Eigenvectors` | Eigenvects square matrix |

## Parallel computation [DC, Tut]

| | |
|---|---|
| `Parallelize[expr]` | Direct parallelization |
| `ParallelEvaluate[expr]` | Eval on kernels |
| `DistributeDefinitions` | Put def on kernels |
| `ParallelNeeds` | Package on kernels |
| `ParallelMap` | Parallel `Map` |
| `ParallelSum` | Parallel `Sum` |
| `ParallelProduct` | Parallel `Product` |
| `ParallelTable` | Parallel `Table` |
| `ParallelDo` | Parallel `Do` |
| `ParallelSubmit` | Submit to kernel<br>Collect with `WaitAll` |
| `WaitAll` | Results from kernels |

## Numerical routines [DC, Tut]

| | | |
|---|---|---|
| `N[expr]` | Numerical eval of `expr` | |
| `SetPrecision[expr,n]` | Precision to $n$ digits | `$Pre=SetPrecision[#,n]&`<br>Set session precision to $n$ digits |
| `NSolve[eqn,var]` | Approx polynomial eqn | |
| `FindRoot[eqn,{x,x0}]` | Approx eqn in var `x` with seed `x0` | |
| `FindRoot[eqn,{x,x0,a,b}]` | Approx `eqn` in $[a,b]$ | |
| `NDSolve[eqn,y,{x,a,b}]` | Solve differential `eqn` | Initial cond specified in `eqn` |
| `NIntegrate[f,{x,a,b}]` | Numerical integration. Methods: | |
| | `"SymbolicProcessing"->0` | Fully numerical eval |
| | `"GlobalAdaptive"` | Default |
| | `"LocalAdaptive"` | |
| | `"Oscillatory"` | 1D integrals; automatic detection |
| | `"PrincipalValue"` | Cauchy principal value |
| | `"MonteCarlo"` | Random sampling |
| | `"AdaptiveMonteCarlo"` | |
| `FindMinimum[f,x,x0]` | Find local min with seed `x0` | |
| `NMinimize[f,x]` | Find global min. Methods: | |
| | `"DifferentialEvolution"` | Option: `"SearchPoints"` |
| | `"NelderMead"` | |
| | `"RandomSearch"` | Option: `"SearchPoints"` |
| | `"SimulatedAnnealing"` | Option: `"SearchPoints"` |

`NumericalCalculus` package:

| | |
|---|---|
| `NLimit[expr,x->x0]` | Numerical limit |
| `ND[expr,x,x0]` | Numerical `D` at `x0` |
| `NSeries[f,{x,x0,n}]` | Numerical series |

## Other

| | |
|---|---|
| `$Pre` | Pre-processing var |
| `$Post` | Post-processing var |

**Some new/updated in Mathematica 8**

| | |
|---|---|
| Free form input | Type = then input |
| `WolframAlpha["query"]` | Query to WolframAlpha server |
| | |
| `RandomVariate[dist]` | Merges `RandomReal` and `RandomInteger` |
| `Probability[pred,x≈dist]` | Probability of predicate with distribution `dist` |
| `NProbability` | |
| `Expectation[expr,x≈dist]` | Expected value with distribution `dist` |
| `NExpectation` | |
| `EstimatedDistribution[data,dist]` | Fit `dist` to `data` |
| `EmpiricalDistribution[data]` | Distribution for `data` |
| `HistorgramDistributon` | Similar to `Histogram` |
| `Distributed[x,dist]` | `x≈dist`. Same as Esc-dist-Esc |
| `Conditioned[expr,cond]` | `expr` conditioned on `cond`. Same as Esc-cond-Esc |
| `Cumulant[dist,r]` | $r^{\text{th}}$ cumulant of `dist` |
| `MomentGeneratingFunction[dist,t]` | Moment function in var $t$ |
| `StableDistribution[type,`$\alpha,\beta,\mu,\sigma$`]` | Stable (Lévy) distribution |
| `TransformedDistribution[trans,dist]` | |
| | |
| `Graph` | |
| `GraphPlot` | |
| `GraphStyle` | |
| `AdjacencyGraph` | |
| `IncidenceGraph` | |
| `GridGraph` | |
| `RandomGraph` | |
| `ButterflyGRaph` | |
| `HighlightGraph` | |
| `AdjacencyMatrix` | |
| `VertexDegree` | |
| `ClosenessCentrality` | |
| `BetweennessCentrality` | |
| `PageRankCentrality` | |
| `GraphQ` | |
| `ConnectedGraphQ` | |
| | |
| `SmoothHistogram` | |
| `SmoothHistogram3D` | |
| `DensityHistogram` | |
| `SmoothDensityHistogram` | |
| | |
| `Texture` | Use image for plot rendering |
| `TextRecognize` | Text recognition from image |
| `Speak` | Speak content |
| `SpokenString` | |
| `ImageCapture[]` | Open camera utility |
| `CurrentImage[]` | Returns current image |
| `EdgeDetect[img]` | Returns image with edges |

**Notes**