Base Gretl-Hansl Cheat Sheet

Getting Help

Accessing the help files

help mean
Get help of a particular function.
help RollingStats
Get help of a particular package.

More about an object

typestr(typeof(LRM))
Find the class an object belongs to.

Using Packages

pkg install RollingStats
Download and install a package from the gretl server.
Include RollingStats.gfn
Load the package into the session, making all its functions available to use.
open denmark.gdt
Load a built-in dataset into the environment.

Working Directory

getwd()
Find the current working directory (where inputs are found and outputs are sent).
setwd('C://file/path')
Change the current working directory.

Use projects in RStudio to set the working directory to the folder you are working in.

The Environment

ls()	List all variables in the environment.
delete x	Remove x from the environment.
clear	Remove all variables from the environment.

Types of objects

string str = "hello world"	string variable
scalar n = 4.3	scalar variable
Series y = 12.0	Series variable
matrix m = ones(10,4)	2-d matrix
list L = y x	List of series
Strings S = defarray("a", "b")	string array
Matrices M = defarray(ones(10,4), zeros(2,4))	Array of 2-d matrices
bundle b = null	bundle object

Matrices

matrix m = ones(3,3)
create a 3 by 3 matrix with ones

m[2,] – select the 2nd row	transp(m) OR m'
m[,1] – select the 1st column	Transpose
m[2,3] – select an element	m * n
	Matrix Multiplication

Series

`if expr: statements`
`elif expr: statements`
`else: statements`
	conditional

`if a is b : ...`	object identity
`if a == 1`	value identity
`while expr: statements` `else: statements`	while loop run else on normal exit
`while True: ... if cond: break`	do... while equivalent
`for target in iter: statements` `else: statements`	for loop
`for key,value in d.items():...`	multiple identifiers
`break, continue`	end loop / jump to next
`print "hello world",`	print without newline
`[ expr for x in seq lc ]` `lc = for x in seq / if expr`	list comprehension with lc-clauses
`pass`	empty statement
`def f(params): statements`	function definition
`def f(x, y=0): return x+y`	optional parameter
`def f(*a1, **a2): statements`	additional list of unnamed, dict of named paramters
`def f(): f.variable = 1 ...`	function attribute
`return expression`	return from function
`yield expression`	make function a generator
`f(1,1), f(2), f(y=3, x=4)`	function calls
`global v`	bind to global variable

`def make_adder_2(a):`
`  def add(b): return a+b`
`  return add`
	closure

`lambda x: x+a`	lambda expression
`compile(string,filename,kind)`	compile string into code object
`eval(expr,globals,locals)`	evaluate expression
`exec code in gldict, lcdict`	compile and execute code
`execfile(file,globals,locals)`	execute file
`raw_input(prompt)`	input from stdin
`input(prompt)`	input and evaluate

3 Object Orientation and Modules

`import module as alias`	import module
`from module import name1,name2`	load attr. into own namespace
`from __future__ import *`	activate all new features
`reload module`	reinitialize module
`module.__all__`	exported attributes
`module.__name__`	module name / "__main__"
`module.__dict__`	module namespace
`__import__("name",glb,loc,fl)`	import module by name
`class name (superclass,...):`	class definition
`  data = value`	shared class data
`  def method(self,...): ...`	methods
`  def __init__(self, x):`	constructor
`  Super.__init__(self)`	call superclass constructor
`  self.member = x`	
`  def __del__(self): ...`	per-instance data destructor
`__str__, __len__, __cmp__,__`	some operator overloaders
`__iter__(self): return self`	use next method for iterator
`__call__`	call interceptor
`__dict__`	instance-attribute dictionary
`__getattr__(self, name),`	get an unknown attribute
`__setattr__(self, name, value)`	set any attribute
`callable(object)`	1 if callable, 0 otherwise
`delattr(object, "name")`	delete name-attr. from object
`del(object)`	unreference object/var
`dir(object)`	list of attr. assoc. with object
`getattr(object, "name", def)`	get name-attr. from object
`hasattr(object, "name")`	check if object has attr.
`hash(object)`	return hash for object
`id(object)`	unique integer (mem address)
`isinstance(object, classOrType)`	check for type
`issubclass(class1, class2)`	class2 subclass of

	class1?
`iter(object, sentinel)`	return iterator for object
`locals()`	dict of local vars of caller
`repr(object), str(object)`	return string-representation
`vars(object)`	return __dict__
`None`	the NULL object
`if __name__ == "__main__":`	make modul executable

4 Exception Handling

`try: ...`	Try-block
`except ExceptionName:`	catch exception
`except (Ex1, ...), data:`	multiple, with data
`    print data`	exception handling
`    raise`	pass up (re-raise) exception
`else: ...`	if no exception occurred
`finally: ...`	in any case
`assert expression`	debug assertion
`class MyExcept(Exception): ...`	define user exception
`raise MyExcept(data)`	raise user exception

5 System Interaction

`sys.path`	module search path
`sys.platform`	operating system
`sys.stdout, stdin, stderr`	standard input/output/error
`sys.argv[1:]`	command line parameters
`os.system(cmd)`	system call
`os.startfile(f)`	open file with assoc. program
`os.popen(cmd, r|w, bufsize)`	open pipe (file object)
`os.popen2(cmd, bufsize, b|t)`	(stdin, stdout) fileobjects
`os.popen3(cmd, bufsize, b|t)`	(stdin, stdout,stderr)
`os.environ['VAR']; os.putenv[]`	read/write environment vars
`glob.glob('*.txt')`	wildcard search

Filesystem Operations

os module: `access, chdir, chmod, chroot, getcwd, getenv, listdir, mkdir, remove, unlink, removedirs, rename, rmdir, pipe, ...`

shutil module: `copy, copy2, copyfile, copyfileobj, copymode, copystat, copytree, rmtree`

os.path module: `abspath, altsep, basename, commonprefix, curdir, defpath, dirname, exists, expanduser, expandvar, extsep, get[acm]time, getsize, isabs, isdir, isfile, islink, ismout, join, lexists, normcase, normpath, pardir, pathsep, realpath, samefile, sameopenfile, samestat, sep, split, splitdrive, splitext, stat, walk`

command line argument parsing:

```
restlist, opts = \
  getopt.getopt(sys.argv[l:],\
    "s:oh",\
    ["spam=", "other", "help"])
for o, a in opts:
    if o in ("-s", "--lol"): spam = a
    if o in ("-h", "--help"): show_help()
```

6 Input/Output

`f=codecs.open(if,"rb","utf-8")`	open file with encoding
`file = open(infilename, "wb")`	open file without encoding
`codecs.EncodedFile(...)`	wrap file into encoding
`r, w, a, r+`	read, write, append, random
`rb, wb, ab, r+b`	modes without eol conversion
`file.read(N)`	N bytes (entire file if no N)
`file.readline()`	the next linestring
`file.readlines()`	list of linestring
`file.write(string)`	write string to file
`file.writelines(list)`	write list of linestrings
`file.close()`	close file
`file.tell()`	current file position
`file.seek(offset, whence)`	jump to file position
`os.truncate(size)`	limit output to size
`os.tmpfile()`	open anon temporary file
`pickle.dump(x, file)`	make object persistent
`x = pickle.load(file)`	load object from file

7 Standard Library (almost complete)

String Services: `string, re, struct, difflib, StringIO, cStringIO, textwrap, codecs, unicodedata, stringprep, fpformat`

File/Directory Access: `os.path, fileinput, stat, statvfs, filecmp, tempfile, glob, fnmatch, linecache, shutil, dircache`

Generic OS services: `os, time, optparse, getopt, logging, getpass, curses, platform, errno, ctypes`

Optional OS services: `select, thread, threading, dummy_thread, dummy_threading, mmap, readline, rlcompleter`

Data Types: `datetime, calendar, collections, heapq, bisect, array, sets, sched, mutex, Queue, weakref, UserDict, UserList, UserString, types, new, copy, pprint, repr`

Numeric and Math Modules: `math, cmath, decimal, random, itertools, functools, operator`

Internet Data Handling: `email, mailcap, mailbox, mhlib, mimetools, mimetypes, MimeWriter, mimify, multifile, rfc822, base64, binhex, binascii, quopri, uu`

Structured Markup Processing Tools: `HTMLParser, sgmllib, htmllib, htmlentitydefs, xml.parsers.expat, xml.dom.*, xml.sax.*, xml.etree.ElementTree`

File Formats: `csv, ConfigParser, robotparser, netrc, xdrlib`

Crypto Services: `hashlib, hmac, md5, sha`

Compression: `zlib, gzip, bz2, zipfile, tarfile`

Persistence: `pickle, cPickle, copy_reg, shelve, marshal, anydbm, whichdb, dbm, gdbm, dbhash, bsddb, dumbdbm, sqlite3`

Unix specific: `posix, pwd, spwd, grp, crypt, dl, termios, tty, pty, fcntl, posixfile, resource, nis, syslog, commands`

IPC/Networking: `subprocess, socket, signal, popen2, asyncore, asynchat`

Internet: `webbrowser, cgi, scitb, wsgiref, urllib, httplib, ftplib, imaplib, nntplib, ...lib, smtpd, uuid, urlparse, SocketServer, ...Server,, cookielib, Cookie, xmlrpclib`

Multimedia: `audioop, imageop, aifc, sunau, wave, chunk, colorsys, rgbimg, imghdr, sndhdr, ossaudiodev`

Tk: `Tkinter, Tix, ScrolledText, turtle`

Internationalization: `gettext, locale`

Program Frameworks: `cmd, shlex`

Development: `pydoc, doctest, unittest, test`

Runtime: `sys, warnings, contextlib, atexit, traceback, qc, inspect, site, user, fpectl`

Custom Interpreters: `code, codeop`

Restricted Execution: `rexec, Bastion`

Importing: `imp, zipimport, pkgutil, modulefinder, runpy`

Language: `parser, symbol, token, keyword, tokenize, tabnanny, pyclbr, py_compile, compileall, dis, pickletools, distutils`

Windows: `msilib, msvcrt, _winreq, winsound`

Misc: `formatter`