GEOMAR
Helmholtz-Zentrum für Ozeanforschung Kiel

Angewandte Programmierübung

Teil III - Ferret

GEOMAR - Helmholtz-Zentrum für Ozeanforschung Kiel

Markus Scheinert*, Christina Roth

Version 1.0.2 (2016)

Inhaltsverzeichnis

*Markus Scheinert, email: mscheinert@geomar.de

Zusätzliche Infos

Beispiele

Tabellen

Copyright

Haftungsausschluss

Notation

In diesem Skript werden bei der Darstellung von Kommandos bzw. Instruktionen Platzhalter der Form "`<Platzhalter>`" verwendet. Dabei muss der Ausdruck in den spitzen Klammern (inklusive der Klammern) durch einen eigenen Ausdruck, z.B. eine Variable oder eine Zuweisung, ersetzt werden. Optionale Angaben sind in großen, grauen, eckigen Klammern gefasst: "`Optional1 [ Optional2 [ ... ] ]`". Stehen mehrere Angaben zur Auswahl, werden sie mit einem Pipe-Zeichen getrennt: "`Opt1|Opt2|Opt3`". Es kann vorkommen, das spitze und eckige Klammern Teil des Kommados sind. Diese werden dann fett gedruckt (**`[ < > ]`**) und meist befindet sich vor und nach diesen Zeichen dann mindestens ein Leerzeichen bzw. im Text wird dann darauf hingewiesen. Ist ein Ausdruck zum Teil unterstrichen (z.B. `show variable`), kann der so markierte Teilausdruck als Abkürzung verwendet werden (`sh va`).

Darüber hinaus werden einige Ausdrücke entsprechend ihrer Bedeutung farblich hervorgehoben: Kommando, KommandoQualifier, Funktion, VariablenQualifier, Transformation und GoSkript.

Teil III

Ferret / PyFerret

1 Einführung

Neben dem orginalen Ferret[1] gibt es mittlerweile auch eine Version für Python[2], **PyFerret** genannt, die auch in diesem Kurs verwendet wird, da sie einige zusätzliche Funktionen bietet. Die in diesem Skript angegebenen Kommandos und Funktionen sind aber, bis auf wenige, gekennzeichnete Ausnahmen, auch mit dem einfachen Ferret verwendbar. Ansonsten wird in diesem Skript "Ferret" gleichbedeutend mit "PyFerret" verwendet und umgekehrt.

1.1 Ferret vs Matlab

- Der größte Unterschied zwischen Matlab und Ferret ist der, dass Matlab ein umfangreiches Analyse- und Visualisierungs-Paket (1D+2D+3D) für Daten unterschiedlichen Ursprungs darstellt, während Ferret hauptsächlich zur Visualisierung (1D+2D) von 'gegitterten' Daten, also Werten auf einem vorgegebenen Raum-Zeit-Gitter, eingesetzt wird.

- In Ferret werden Beziehungen zwischen Variablen und Werten zunächst definiert und erst dann ausgewertet, wenn ein Plot erstellt wird oder konkret das Ergebnis einer Berechnung abgefragt wird. Matlab hingegen arbeitet hauptsächlich mit Kommandos, die sofort ausgeführt werden. Das hat aber auch Konsequenzen für den Computer-Speicher.

- Werden in Ferret Daten z.B. im NetCDF-Format genutzt (ein spezielles Format für Daten auf einem festen Gitter, z.B. Länge, Breite, Tiefe/Höhe und Zeit), dann werden nur die Werte in den Speicher geladen, die wirklich notwendig sind, z.B. eine Region, eine Tiefe oder ein Zeit-Ausschnitt. Matlab versucht zumeist den gesamten Datensatz einzulesen. Das kann bei großen Datensätzen von mehreren Giga-Byte eine wichtige Rolle spielen.

- Ferret ist frei verfügbar wohingegen Matlab ein kommerzielles Produkt ist.

Faszit:

- Werden 'ungegitterte' Daten verwendet oder sollen umfangreiche 2D/3D Darstellungen erzeugt werden, ist Matlab vermutlich die bessere Wahl.
- Stehen die Daten auf einem festen Ort-Zeit-Gitter zur Verfügung (wie z.B. Modell-Daten) oder sind die Daten zu umfangreich, dann können mit Ferret schnell und effektiv Illustrationen erzeugt werden.

1.2 Environment

Die Informationen, wo auf dem Rechner Skripte, Farb-Paletten und Datensätze stehen, nimmt Ferret aus UNIX-Umgebungsvariablen, die mittels zentral abgelegter oder persönlicher Konfigurationsdateien (*ferret_paths*) festgelegt werden. In Fall dieses Kurses liegen sie im HOME-Verzeichnis (~) unter `~/RC/` :

[1] http://ferret.pmel.noaa.gov/Ferret/home
[2] https://www.python.org/

csh/tcsh:

```
> source ~/RC/pyferret_paths.csh
```

sh/bash/ksh:

```
> . ~/RC/pyferret_paths.sh
```

Diese Kommandos (das 'Punkt'-Kommando bei sh/bash/ksh bzw. 'source' in der csh/tcsh) lesen die Definitionen für verschiedene Umgebungsvariablen für Ferret ein und machen sie in der aktuellen Shell wirksam, als ob sie direkt eingetippt würden. Würde man die Skripte nur ausführen würde eine Subshell gestartet werden, in dieser dann die Umgebungsvariablen gesetzt und die Shell wieder verlassen werden. Die zuvor gesetzten Variablen wären aber hinfällig (siehe auch Info-Box 1).

Info 1: UNIX Umgebungsvariablen

Programme unter Unix/Linux nehmen häufig Einstellungen und Such-Pfade für weitere Dateien aus Umgebungsvariablen der Shell, in der das Programm aufgerufen wird. Da man beliebig viele Shelles ineinandergeschachtelt starten kann, ist es wichtig auf welcher Stufe Umgebungsvariablen gesetzt oder verändert werden:

```
>                                              Nach dem Login
> setenv stufe0 Hallo
> echo $stufe0
Hallo
> csh                                          Starte 1. Sub-Shell


    This is csh...
    > echo $stufe0
    Hallo
    > setenv stufe0 Halloechen
    > setenv stufe1 Du
    > csh                                      Starte 2. Sub-Shell


        This is csh...
        > echo $stufe0 $stufe1
        Halloechen Du
        > setenv stufe2 Eumel
        > echo $stufe0 $stufe1 $stufe2
        Halloechen Du Eumel
        > exit
        exit                                   2. Sub-Shell beendet

    > echo $stufe0 $stufe1 $stufe2
    stufe2: Undefined variable.                Variable aus 2. Subshell existiert hier
    nicht.
    > echo $stufe0 $stufe1
    Halloechen Du
    > exit
    exit                                       1. Sub-Shell beendet


> echo $stufe0
Hallo                                          Variable hat wieder den alten Wert
>
```

Danach werden Umgebungsvariablen auf Sub-Shells zwar vererbt, außerhalb bleiben sie jedoch unverändert bzw. werden neue Variablen nicht an die 'Eltern-Shell' übertragen.

1.3 Ferret starten

Nach dem Einladen der Umgebungsvariablen wird PyFerret über das Unix-Kommando `pyferret` ↵ [3] gestartet und meldet sich dann mit einer eigenen Eingabeaufforderung, dem Prompt: `yes?`:

```
> pyferret
        NOAA/PMEL TMAP
        FERRET v6.9 (PyFerret 1.0.2)
        Linux 3.5.0-48-generic - 04/01/14
        24-Oct-14 10:31
yes?
```

Ein zusätzliches `yes?` am Anfang der Zeile wird übrigens von Ferret nicht interpretiert, was praktisch ist, wenn man Zeilen beispielsweise mit der Maus markiert und einfügt[4]. Im Hintergrund protokolliert Ferret alle Eingaben in der Datei `ferret.jnl` mit (Existiert schon ein `ferret.jnl`, wird dieses umbenannt in `ferret.jnl~x~`, wobei `x` von 1 an hochgezählt wird). Wird dies nicht gewünscht, kann das mit der Option `-nojnl` beim Programmaufruf unterbunden werden (siehe auch Info 2):

```
> pyferret -nojnl
```

Info 2: PyFerret Kommandozeilen-Optionen:

```
> pyferret --help

  Usage:  pyferret  [-memsize <N>]  [-nodisplay]  [-nojnl]  [-noverify]
                      [-secure]  [-server]  [-python]  [-version]  [-help]
                      [-quiet]  [-batch [<filename>]]  [-transparent]
                      [-script <scriptname> [ <scriptarg> ... ]]

       -memsize:     initialize the memory cache size to <N> (default 25.6)
                     mega (10^6) floats (where 1 float = 8 bytes)
                     {25.6x10^6 floats ~ 195 Mb}
       -nodisplay    do not display to the console; a drawing can be saved
                     using the FRAME command in any of the supported file
                     formats.  The /QUALITY option of SET WINDOW will be
                     ignored when this is specified.  The deprecated
                     command-line options -unmapped and -gif are now
                     aliases of this option.
       -nojnl:       on startup do not open a journal file (can be turned
                     on later with SET MODE JOURNAL)
       -noverify:    on startup turn off verify mode (can be turned on
                     later with SET MODE VERIFY)
       -secure:      restrict Ferret's capabilities (e.g., SPAWN and
                     EXIT /TOPYTHON are not permitted)
       -server:      run Ferret in server mode
       -python:      start at the Python prompt instead of the Ferret prompt.
                     The ferret prompt can be obtained using 'pyferret.run()'
       -version:     print the Ferret header with version number and quit
       -help:        print this help message and quit
       -quiet        do not display the startup header or
                     warning of import failures
       -batch:       draw to <filename> (default "ferret.png") instead of
                     displaying to the console.  The file format will be
                     guessed from the filename extension.  When using this
                     option, new windows should not be created and the
                     FRAME command should not be used.
                     Use of -batch (and -transparent) is not recommended.
                     Instead use the -nodisplay option and the FRAME
                     /FILE=<filename> [ /TRANSPARENT ] command.
       -transparent: use a transparent background instead of opaque white
                     when saving to the file given by -batch
       -script:      execute the script <scriptname> with any arguments specified,
                     and exit (THIS MUST BE SPECIFIED LAST)
```

[3] ↵: Taste 'Enter' oder 'Return' oder auch 'Eingabe'.

[4] TIPP: Mit gedrückter linker Maustaste Text markieren, linke Maustaste loslassen, den Zeiger an die gewünschte Stelle versetzen und mittlere Maustaste (oder das Scroll-Rad) kurz drücken. Dann wird der eben markierte Text aus dem "Maus-Puffer" an die gewünschte Stelle eingefügt.

2 Ferret Grundlagen

2.1 Kommandos & Scripting

Kommandos in Ferret haben die Form[5]:

```
yes? Kommando/Qualifier Ausdruck
```

`yes?`	Prompt/Eingtabeaufforderung von Ferret; gehört nicht zum Kommando.
Kommando	kann aus einem oder zwei Wörtern bestehen
/Qualifier	beeinflusst das Kommando in eindeutiger Weise. Evt. sind zusätzliche Angaben der Form "`/Qualifier=Wert`" nötig.
Ausdruck	kann eine Variable sein oder ein arithm. Ausruck

Sämtliche Kommandos und Variablen Namen sind nicht von der Schreibweise (Groß-/Kleinbuchstaben) abhängig.

Ein typische Beispiel für eine Kommandofolge in Ferret wäre die Darstellung der potentiellen Temperatur (temp) aus der sog. COADS-Klimatologie, die als Datensatz mit Ferret zusammen kommt, z.B. bei 500m Tiefe:

```
yes? set data/format=cdf coads_climatology
yes? shade/levels=(0,30,2)/nolabel temp[z=500]
```

Das gleiche Beispiel nur mit verkürzten Befehlen (siehe unten) und einer Kommando-Ersetzung (Alias 'use') für das Einladen des Datensatzes:

```
yes? use coads_climatology
yes? sha/lev=(0,30,2)/nol temp[z=500]
```

Häufig benötigte Kommandos

Kommando	Verk. Kom.	Bedeutung
`!`		Rest der Zeile ist Kommentar
`show data/all`	`sh da/a`	Listet alle geladenen Datensätze auf.
`set data 1`	`se da 1`	Setzt den ersten Datensatz als Default-Datensatz
`let VAR1 = VAR2`		Zuweisungs-Operator für Variable `VAR`
`show variable VAR`	`sh va VAR`	Ausgabe der Definition von `VAR`
`list VAR`	`li VAR`	Auswertung und Ausgabe der/s Variablen-Werte/s von `VAR`
`show grid VAR`	`sh gr VAR`	Ausgabe der Gitter-Informationen für `VAR`
`spawn ls`	`sp ls`	Führt das UNIX Kommando `ls` aus, ohne Ferret zu verlassen
`quit`	`q`	Beendet Ferret (alternativ auch: `exit`)

Eine vollständige Kommandoübersicht befindet sich im Anhang A auf Seite 73.

[5] *Die in diesem Skript unterstrichenen Buchstaben der Befehle und Qualifier können als Abkürzung verwendet werden (siehe Paragraph Kommando-Abkürzung)*

Kommando-Abkürzung

Kommandos können in Ferret abgekürzt werden, solange sie eindeutig bleiben, also zum Beispiel:

```
sh va VAR
```

statt

```
show variable VAR
```

aber nicht

```
s v VAR ,
```

denn dass könnte auch `set variable VAR` sein, eine ausführliche Variante des Kommandos `let`. **Variablen-Namen** hingegen müssen immer komplett genannt werden.

Alias

Einige Kommandos, die ursprünglich recht lang sind, weil einige Qualifier verwendet werden, können durch einen Alias-Befehl ersetzt werden (siehe Info 3) der wiederum unter Umständen weiter abgekürzt werden kann.

Info 3: Kommando-Ersetzungen (Alias):

Das Kommando `show alias` listet die aktuell definierten Kommando-Ersetzungen auf:

```
yes? show alias

   Alias       Command
   -----       -------
   LET         DEFINE VARIABLE
   FILE        SET DATA/EZ
   QUIT        EXIT
   REGION      SET REGION
   SAY         MESSAGE/CONTINUE
   FILL        CONTOUR/FILL
   ALIAS       DEFINE ALIAS
   UNALIAS     CANCEL ALIAS
   USE         SET DAT/FORM=CDF
   SAVE        LIST/FORMAT=CDF
   PALETTE     PPL SHASET SPECTRUM=
   LABEL       PPL %LABEL
   ANIMATE     REPEAT/ANIMATE
   PATTERN     PPL PATSET PATTERN=
   PAUSE       MESSAGE
   WHERE       PPL %WHERE
   FLOWLINE    VECTOR/FLOWLINE
   COLUMNS     SET DATA/FORM=delimited
   KEYMARK     PPL SHAKEY 1,,,,,,,,,,
   OPEN        SET DATA /BROWSE
   RIBBON      PLOT/RIBBON
   ENSEMBLE    DEFINE DATA/AGGREGAT/E
   VTREE       SHOW VAR/TREE
```

`set` vs. `cancel`

Einige Kommandos, die mit `set` oder `define` beginnen, wie *`set region`* oder *`define variable`*, haben ein entsprechendes `cancel` Kommando, um dieses wieder aufzuheben.

Scripting

Ferret Skripte enthalten Ferret-Kommandos in der Reihenfolge, wie sie auch am Prompt vom Benutzer eingegeben würden. Bestimmte Abläufe lassen sich so in eine Art Programm abspeichern und innerhalb von Ferret wieder ausführen (ähnlichen den Matlab-Scripten oder Shell-Skripten). In Ferret haben solche Skripte die Dateinamserweiterung **.jnl** und werden am ehesten im aktuellen Verzeichnis (in dem ferret gestartet wurde) gefunden. Die Skripte selbst lassen sich mittels go und dem Dateinamen des Skriptes (mit oder ohne '.jnl') ausführen:

```
go <SkriptName> [ <arg1> [ <arg2> [ ... ] ] ]
```

<arg1>, <arg2>,... usw. sind Argumente, die dem Skript übergeben werden können; innerhalb des Sktiptes kann man dann über die Variablen $1, $2,... usw. darauf zugreifen.

Wichtig ist jedoch, daran zu denken, dass die Anweisungen nicht strikt konsekutiv sein müssen und Variablen und Ausdrücke erst ausgewertet werden, wenn das Ergebnis z.B. zum Plotten angefordert wird (ein gravierender Unterschied zu Matlab!).

Manche Skripte enthalten in den ersten Zeilen nützliche Hinweise und Hilfen zum Umgang mit dem Skript (z.B. welche Agrumente das Skript erwartet und Anwendungsbeispiele). Um einen Blick in das Skript selbst werfen zu können, gibt es das Kommando go/help <SkriptName> (Zum Verlassen Taste q drücken).

Info 4: Ferret Go-Skripte:

Ferret kommt bereits mit einer Vielzahl von Skripten, mit deren Hilfe Daten manipuliert werden können oder eben bestimmte Plot-Befehle ausgeführt werden können. Eine Liste aller Go-Skripte lässt sich mit dem UNIX-Kommando Fgo erzeugen (siehe auch Anhang D):

```
> Fgo '*'
```

2.2 Datensätze

Ferret kann unterschiedliche Datensätze lesen, am einfachsten geht es jedoch mit sog. NetCDF-Daten (Endung .nc oder .cdf) und dem Kommando **use**, z.B.:

```
yes? use coads_climatology
```

Die Pfade, die Ferret nach der angegebenen Datei durchsucht, sind das lokale Arbeitsverzeichnis und die Verzeichnisse in der Umgebungsvariablen $FER_DATA. Zu letzterem gehört auch das Beispiel-Daten-Verzeichnis von Ferret, in dem die COADS Klimatologie abgelegt ist. Konnte ein Datensatz erfolgreich geladen werden, fügt Ferret diesen einer Liste der eingeladenen Dateien an, beginnend mit der Nummer 1. Der zuletzt eingeladene Datensatz ist immer der **default-Datensatz** es sei denn er wird mit **set data <Datensatz-Nummer>** geändert

Die Liste aller geladenen Dateien lässt sich mit Hilfe des Kommandos show data einsehen. Alternativ kann man sich eine Kurz-Liste (Schalter /brief) nur mit den Dateinamen und Pfaden ausgeben lassen oder eine erweiterte Liste mit Angaben zu Einheiten und Missing-Values (Schalter /full; siehe Beispiel Beispiel 1).

Es ist auch möglich, ASCII-Daten (z.B. CSV) einzulesen, erfordert jedoch etwas mehr Wissen über Axen, Gitter und Zahlenformate und wird daher später behandelt.

Beispiel 1

```
>pyferret
        NOAA/PMEL TMAP
        FERRET v6.7
        Linux(gfortran) 2.6.18-238.1.1.el5 - 05/06/11
        24-Oct-11 17:29

yes? use coads_climatology

yes? sh da/b
     currently SET data sets:
    1> ./coads_climatology.cdf

yes? sh da
     currently SET data sets:
    1> ./coads_climatology.cdf
 name     title                          I         J         K         L
 SST      SEA SURFACE TEMPERATURE        1:180     1:90      ...       1:12
 AIRT     AIR TEMPERATURE                1:180     1:90      ...       1:12
 SPEH     SPECIFIC HUMIDITY              1:180     1:90      ...       1:12
 WSPD     WIND SPEED                     1:180     1:90      ...       1:12
 UWND     ZONAL WIND                     1:180     1:90      ...       1:12
 VWND     MERIDIONAL WIND                1:180     1:90      ...       1:12
 SLP      SEA LEVEL PRESSURE             1:180     1:90      ...       1:12
```

2.3 Variablen

Es gibt *Datei-Variablen* und *benutzerdefinierte* Variablen. Erstere werden mit einem Datensatz eingeladen und sind in der NetCDF-Datei festgelegt oder werden mit dem Lese-Kommando festgelegt. Die anderen Variablen werden vom Benutzer definiert und beschreiben die formale Beziehung verschiedener Größen/Variablen oder beinhalten Werte, die vom Benutzer direkt eingegeben wurden, etwa als Parameter.

Eigene Variablen definieren

Eigene Variablen können mit dem Kommando `let`[6] definiert werden:

`let/<Qualifier> <VARIABLE> = <AUSDRUCK>`

Variablen-Namen bestehen aus einer alpha-numerischen Zeichenfolge, beginnend mit einem Buchstaben, und können einzig **Unterstriche** als Sonerzeichen enthalten. Groß-/Kleinschreibung wird nicht unterschieden. Einige Beispiele:

Richtig	Falsch	Erläuterung
`sst`	`s.s.t`	Punkte (wie Kommas und Semikolons) sind nicht erlaubt
`meine_variable`	`Meine-Variable`	Minus-Zeichen ist nicht erlaubt
`z33_05`	`33z_05`	Die Variable muss mit einem Buchstaben beginnen.

Einer Variablen kann z.B. auch ein Titel und eine Einheit mitgegeben werden, die dann statt der Variablen-Definition im Plot angezeigt werden:

```
let/title="One above Absolute Zero"/units="Kelvin" a = b+1
```

Eine Auflistung möglicher Kommando-Qualifier zeigt Tabelle 1.

[6] Genauer handelt es sich bei `let` um ein sog. alias für das Kommando `set variable`.

Kommando-Qualifier	Erläuterung
/title	Langer Variablen-Name
/units	Physikalische Einheit der Variable
/bad=	Legt fest, welcher Wert als 'Missing Value' behandelt werden soll (siehe auch Seite 35)
/D=	Legt die Datei als Datensatzbezogene Variable fest

Tabelle 1: Kommando-Qualifier für `define variable`.

Datensatzabhängige Variablen Besonders vorzuheben ist der Qualifier /D=, mit dessen Hilfe Variablen nur für einen bestimmten Datensatz definiert werden können. Das ist nützlich, wenn in einem oder mehr Datensätzen eine Variable bereits enthalten ist, die man für andere Datensätze erst definieren muss. Zum Beispiel enthält der COADS-Datensatz eine Variable `sst`, die im Levitus-Datensatz nicht vorhanden ist, aber leicht aus der 3-dimensionalen Größe `temp` erzeugt werden kann. Will man jedoch eine solche Variable `sst` dann für COADS verwenden, gibt es einen Konflikt, weil diese ja schon existiert. Lösung: `sst` wird nur für Levitus eingeführt (siehe Beispiel 2).

Beispiel 2

```
yes? use levitus_climatology
yes? use coads_climatology
yes? sh da
     currently SET data sets:
    1> /usr/local/ferret/fer_dsets/data/levitus_climatology.cdf
 name     title                          I         J         K         L
 TEMP     TEMPERATURE                 1:360     1:180     1:20      ...
 SALT     SALINITY                    1:360     1:180     1:20      ...

    2> /usr/local/ferret/fer_dsets/data/coads_climatology.cdf  (default)
 name     title                          I         J         K         L
 SST      SEA SURFACE TEMPERATURE     1:180     1:90      ...       1:12
 AIRT     AIR TEMPERATURE             1:180     1:90      ...       1:12
 SPEH     SPECIFIC HUMIDITY           1:180     1:90      ...       1:12
 WSPD     WIND SPEED                  1:180     1:90      ...       1:12
 UWND     ZONAL WIND                  1:180     1:90      ...       1:12
 VWND     MERIDIONAL WIND             1:180     1:90      ...       1:12
 SLP      SEA LEVEL PRESSURE          1:180     1:90      ...       1:12

yes? let/d=1 sst=temp[k=1]
yes? sh da
     currently SET data sets:
    1> /usr/local/ferret/fer_dsets/data/levitus_climatology.cdf
 name     title                          I         J         K         L
 TEMP     TEMPERATURE                 1:360     1:180     1:20      ...
 SALT     SALINITY                    1:360     1:180     1:20      ...
 ------------------------------
 SST[D=levitus_climatology] = TEMP[K=1]

    2> /usr/local/ferret/fer_dsets/data/coads_climatology.cdf  (default)
 name     title                          I         J         K         L
 SST      SEA SURFACE TEMPERATURE     1:180     1:90      ...       1:12
 AIRT     AIR TEMPERATURE             1:180     1:90      ...       1:12
 SPEH     SPECIFIC HUMIDITY           1:180     1:90      ...       1:12
 WSPD     WIND SPEED                  1:180     1:90      ...       1:12
 UWND     ZONAL WIND                  1:180     1:90      ...       1:12
 VWND     MERIDIONAL WIND             1:180     1:90      ...       1:12
 SLP      SEA LEVEL PRESSURE          1:180     1:90      ...       1:12
```

2.4 Gitter und Achsen

Variablen in Ferret sind immer mit einem Raum-Zeit-Gitter verknüpft. Auf welchem Gitter eine Variable gerade definiert ist und welche Eingenschaften die jeweiligen Achsen haben, verrät der Befehl:

```
show grid variable
```

Dabei wird der Name des Gitters (im Beispiel Beispiel 3: `GSQ1`) ausgegeben und die Namen, die Achsen-Typen, die Anzahl und Qualitäten der Gitterpunkte (r='regular', i='irregular', m='modulo') sowie Start- und Endwerte.

Achsen-Typen

Ferret definiert Variablen auf bis zu 6 verschiedenen Achsen:

Kontext	Richtung/Achse	Index	Typ
Raum	X	I	Normal/Abstract/Metrisch/Längengrad
	Y	J	Normal/Abstract/Metrisch/Breitengrad
	Z	K	Normal/Abstract/Metrisch/Tiefe
Zeit	T	L	Normal/Abstract/Zeit
Ensemble	E	M	Normal/Abstract/Metrisch/Längengrad/Zeit
Forecast	F	N	Normal/Abstract/Zeit

- Jede Achse (X,Y,Z,T,E,F) ist mit einem entsprechenden Index (I,J,K,L,M,N) verknüpft. Erstere (Achsen) können eine physikalische Einheit und Dezimalzahlen als Werte haben während letztere (Indizes) grundsätzlich ganzzahlig sind und keine Einheit haben.
- Achsen-Typ **'Normal'** gibt an, dass die Variable auf dieser Achse nicht explizit definiert ist bzw. 'normal' auf dieser Achse steht (Index=1).
- Achsen-Typ **'Abstract'** heißt, es gibt keine besonderen Achsen-Werte und Einheiten bzw. die Achsen-Punkte werden nur durchgezählt (entspricht dann dem Index beginnend mit 1).
- **Metrische** Achsen-Typen haben die Längeneinheit Meter, wobei eine explizite Tiefenachse positive Werte nach unten gerichtet hat.
- **Längen**grade können als positive 360°-Werte nach Osten gerichtet angegeben werden oder aber in positiver Ost- und negativer West (W)-Richtung. Bei Modulo-Achsen können sogar werte >360° angegeben werden.
- Die **Zeit** kann unterschiedliche Einheiten haben, wie z.B. Sekunden, Tage oder auch Jahre. Dabei ist jedoch der verwendete Kalender wichtig (Schaltjahre (**leap**), keine Schaltjahre (**noleap**), Monate mit gleicher Länge...). Die Zeit hat auch meistens eine 24h-Uhrzeit-Angabe, also: "01-JAN-2007 12:00:00".
- Achsen können als sich wiederholende **Modulo**-Achsen definiert sein, d.h. am Ende der Achse schließt sich der erste Wert der Achse wieder an (z.B. ein klimatologisches Jahr).
- Die Achsen E und F sind spezielle Achsen, die dazu genuzt werden können **Ensemble-Mitglieder** (auf einer Raum- oder Zeitachse) zusammen zu fassen (typischerweise über die Achse E) oder **Forecast-Zeiten** (über Achse F jedoch nur als Zeit-Achse) zu sortieren.

Eine Variable ist nie nur mit einer oder einigen wenigen Achsen verknüpft sondern immer mit einem vollständigen Satz von X, Y, Z, T, E, F-Achsen, bzw. genauer, mit einem benannten Gitter, das diesen Achsen-Satz zusammenfasst.

Ferret führt seine Rechenoperationen entlang der jeweiligen Achse unter Berücksichtigung deren Einheit aus. So ergibt z.B. das Integral einer Geschwindigkeits-Variablen (Einheit m/s) entlang der X-Achse (vom Typ Longitude) die Einheit m^2/s, weil Ferret den Abstand (in m) zwischen zwei Grad-Angaben auf einem typischen Mercator-Gitter kennt. Aber Vorsicht: Ferret passt nicht das Label der Einheiten an! Beim Plotten wird daher m/s an der Y-Achse des Integrals stehen.

Ohne weitere Achsen-Angaben werden Variablen auf dem Gitter seiner Quell-Größen (erste Variable auf der rechten Seite der Variablen-Zuweisung) definiert oder, wenn Daten direkt eingegeben werden, auf einer Achse *Abstract* bzw. *Normal*.

Beispiel 3

```
yes? use coads_climatology
yes? sh gr sst
    GRID GSQ1
 name       axis              # pts   start               end
 COADSX    LONGITUDE          180mr   21E                 19E(379)
 COADSY    LATITUDE            90 r   89S                 89N
 normal    Z
 TIME      TIME                12mr   16-JAN 06:00        16-DEC 01:20
 normal    E
 normal    F

yes? sh ax time
 name       axis              # pts   start               end
 TIME      TIME                12mr   16-JAN 06:00        16-DEC 01:20

yes? sh ax/t time
 name       axis              # pts   start               end
 TIME      TIME                12mr   16-JAN 06:00        16-DEC 01:20
T0 = 01-JAN-0000 00:00:00
   Axis span (to cell edges) = 8765.82 (modulo length = axis span)

       L     T                   TBOX      TBOXLO                TSTEP (hour)
       1>  16-JAN 06:00:00       730.485    01-JAN 00:45:27         366
       2>  15-FEB 16:29:06       730.485    31-JAN 11:14:33         1096.485
       3>  17-MAR 02:58:12       730.485    01-MAR 21:43:39         1826.97
       4>  16-APR 13:27:18       730.485    01-APR 08:12:45         2557.455
       5>  16-MAY 23:56:24       730.485    01-MAY 18:41:51         3287.94
       6>  16-JUN 10:25:30       730.485    01-JUN 05:10:57         4018.425
       7>  16-JUL 20:54:36       730.485    01-JUL 15:40:02         4748.91
       8>  16-AUG 07:23:42       730.485    01-AUG 02:09:08         5479.395
       9>  15-SEP 17:52:47       730.485    31-AUG 12:38:14         6209.88
      10>  16-OCT 04:21:53       730.485    30-SEP 23:07:20         6940.365
      11>  15-NOV 14:50:59       730.485    31-OCT 09:36:26         7670.85
      12>  16-DEC 01:20:05       730.485    30-NOV 20:05:32         8401.335

yes? sh ax/x coadsx
name       axis              # pts   start               end
COADSX    LONGITUDE          180mr   21E                 19E(379)
Axis span (to cell edges) = 360 (modulo length = axis span)
       I     X                   XBOX      XBOXLO
       1>  21E                   2          20E
       2>  23E                   2          22E
         ...
      79>  177E                  2          176E
      80>  179E                  2          178E
      81>  179W                  2          180E
      82>  177W                  2          178W
         ...
     169>  3W                    2          4W
     170>  1W                    2          2W
     171>  1E(361)               2          0E(360)
     172>  3E(363)               2          2E(362)
         ...
     179>  17E(377)              2          16E(376)
     180>  19E(379)              2          18E(378)
```

2.5 Variablen-Qualifier

Zu jeder Variable kann man noch sog. Variablen-Qualifier in eckigen Klammern **[ ]** mit angeben, um zum Beispiel ein bestimmtes Element der Variable anzusprechen, die Achse bzw. das gesamte Gitter zu verändern oder auf einen bestimmten Datensatz zuzugreifen:

***Variable*[<qualifier> = <sepcification>** `,...` **]**

Zum Beispiel wird mittels `sst[i=2,g=sss,d=2]` auf das zweite Element der X-Achse der Variable `sst` vom 2. Datensatz zugegriffen, wobei die Variable (linear) auf das Gitter von `sss` interpoliert wird (siehe Kapitel 5).

Zugriff auf einzelne Elemente oder Bereiche einer Variablen

Da jede Variable von vornherein mit dem Raum-Zeit-Gitter verküpft ist, kann jedes einzelne Element über einen entsprechenden Index oder den verküpften Achsen-Wert als Qualifier angesprochen werden (*Variable*[$\mathcal{X}$ =...]). Es gibt drei Varianten:

$\mathcal{X}$ = <EINZELPUNKT>
$\mathcal{X}$ = <START>:<END>
$\mathcal{X}$ = <START>:<END>:<DELTA>

Dabei steht $\mathcal{X}$ entweder für eine Achse (X,Y,Z,T,E,F) oder einen Index (I,J,K,L,M,N). Einzelpunkt-Angaben (nur ein Wert an Stelle des Platzhalters <EINZELPUNKT>) sind genauso gültig, wie Bereichsangaben mit Werten zwischen <Start> und <END>. Bei letzerem kann auch eine Schrittweite <DELTA> mit angegeben werden; wird sie weggelassen, wird jeder Punkt der Achse berücksichtigt. Eine Übersicht bietet Tabelle 2. Eine Festlegung der gültigen Achsenabschnitte als Variablen-Qualifier ist natürlich nur für die jeweilige Variable gültig. Alterantiven werden später im Abschnitt 2.7 besprochen. Darüber hinaus sind sog. Achsen-Transformationen möglich (Kapitel 4.2).

Index	Achse	Beispiel	Erläuterung
i	x	i=1	erstes Element auf der X-Achse
		i=1:10:2	jedes zweite Element der X-Achse angefangen mit dem ersten
		x=40W:10E	Alle Elemente zwischen 40° West und 10°E
j	y	j=1	wie bei X
		y=10S:10N:2.5	Jedes Element zwischen 10°S und 10°N alle 2.5°
k	z	k=10	wie X
		z=-300:-800	Alle Elemente zwischen 800 und 300 (m) Tiefe
l	t	l=12	Der 12. Zeitpunkt
		t="01-JAN-1948 12:00:00":"03-JUN-1952 08:30:00"	Alle Elemente im angegebenen Zeitraum
e	m	m=16	Das 16. Ensemble-Mitglied
f	n	f="1-jan-1980":"31-jan-1980"	Alle Vorhersagen im Januar 1980
@	@	@NATL	Alle Axen auf die Region “NATL” einschränken

Tabelle 2: Variablen-Qualifier 1: Zugriff auf Elemente.

Zugriff auf Gitter, Axen und Datensätze

Nicht nur Achsen-Abschnitte können mit Variablen-Qualifiern angesprochen werden, sondern auch der damit verknüpfte Datensatz oder das Gitter bzw. die einzelnen Achsen (siehe Tabelle 3). Das kann man nutzen, um z.B. Achsen- oder Gittertransformationen (siehe Kapitel 5.2) auszuführen, und so zwischen verschiedenen Achsen/Gittern zu interpolieren.

Komponente	Beispiel	Erläuterung
g	g=sst	Verwende das Gitter, auf dem die Variable sst definiert ist.
	g=gsx1	Verwende das Gitter gsx1.
gx	gx=sst	Verwende die X-Achse auf der auch die Variable sst definiert ist.
gy	gy=mylat	Verwende die Y-Achse mylat.
gz	gz=regdepth@LIN	Verwendet explizit eine lineare Interpolation[7] auf die neue Z-Achse regdepth.
gt		
ge		
gf		
d	d=1	Verwende den Datensatz mit der Nummer 1.
	d=coads_climatology	Verwende den Datensatz mit dem Namen coads_climatology.nc

Tabelle 3: Variablen-Qualifier 2: Verweise auf Achsen und Datensätze.

2.6 Pseudo-Variablen

Die Werte der sechs Achsen-Richtungen können über die **Pseudo-Variablen** _X, _Y, _Z, _T, _E, _F bzw. ihre Indizes (_I, _J, _K, _L, _M, _N) angesprochen werden (vgl. Info-Box 5). Der Abstand zwischen zwei Gitterpunkten ist in den Pseudo-Variablen *BOX abgelegt, die Unter (Ober) -Grenzen der Boxen in *BOXLO (*BOXHI).

Achsen-wert[8]	Index[8]	Box-breite	Wert des unteren Boxendes	Wert des oberen Boxendes
_X ,X	_I, I	XBOX	XBOXLO	XBOXHI
_Y, Y	_J, J	YBOX	YBOXLO	YBOXHI
_Z, Z	_K, K	ZBOX	ZBOXLO	ZBOXHI
_T, T	_L, L	TBOX	TBOXLO	TBOXHI
_E	_M	EBOX	EBOXLO	EBOXHI
_F	_N	FBOX	FBOXLO	FBOXHI

[7] @LIN ist eine sog. *Regridding*-Funktion. Diese fortgeschrittene Technik sei hier nur erwähnt und wird später erläutert.

Zum Beispiel liefert `list _X[i=2,g=sst]` den 2. X-Wert auf der Achse der Variable `sst` und `list _K[z=4200, g=temperature]` liefert den K-Index für eine Tiefe von 4200m der Variable `temperature`.

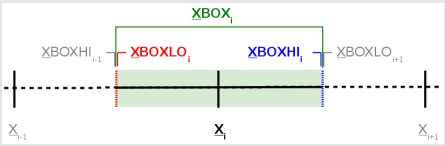
Info 5: Achsenwerte & Pseudovariablen

Die Pseudovariablen von Ferret verdeutlichen, wie Punkte auf der Achse verteilt sind:

Pseudo-Variable	Achsen-Äquivalent
_$\mathcal{X}$	Achsen-Wert in der Mitte der Box
$\mathcal{X}$BOX	Breite der Box auf der Achse
$\mathcal{X}$BOXLO	Achsen-Wert des linken Boxandes
$\mathcal{X}$BOXHI	Achsen-Wert des rechten Randes

$\mathcal{X}$ steht dabei für eine der vier Achsen X, Y, Z, T, E, F.

Dabei liegt ein bestimmter Achsen-Punkt in der Mitte einer gedachten Box von bestimmter Breite. Reguläre Achsen haben überall die gleiche Box-Breite, irreguläre nicht.



Die tatsächlichen Werte kann man sich auch ausgeben lassen mit Hilfe des Kommandos `list`, den Pseudovariablen und entsprechenden Qualifiern für die Achsen (entweder direkt über den entsprechende Achsen-Name (`gx, gy, gz, gt, ge, gf`) oder, allgemeiner, den Gitter-Namen (`g`)):

```
yes? use coads_climatology
yes? sh gr sst

    GRID GSQ1
 name       axis              # pts   start               end
 COADSX     LONGITUDE         180mr   21E                 19E(379)
 COADSY     LATITUDE           90 r   89S                 89N
 normal     Z
 TIME       TIME               12mr   16-JAN 06:00        16-DEC 01:20
 normal     E
 normal     F

yes? list XBOXLO[gx=coadsx],X[gx=coadsx],XBOXHI[gx=coadsx],XBOX[gx=coadsx], XBOX[g=gsq1]

            LONGITUDE: 20E to 20E(380)

 Column  1: XBOXLO is XBOXLO (axis COADSX)
 Column  2: X is X (axis COADSX)
 Column  3: XBOXHI is XBOXHI (axis COADSX)
 Column  4: XBOX is XBOX (axis COADSX)
 Column  5: XBOX is XBOX (axis COADSX)

              XBOXLO     X   XBOXHI  XBOX   XBOX
21E    /   1:   20.0   21.0   22.0  2.000  2.000
23E    /   2:   22.0   23.0   24.0  2.000  2.000
25E    /   3:   24.0   25.0   26.0  2.000  2.000
27E    /   4:   26.0   27.0   28.0  2.000  2.000
29E    /   5:   28.0   29.0   30.0  2.000  2.000
31E    /   6:   30.0   31.0   32.0  2.000  2.000
```

[8] Aus Kompatibilitätsgründen mit alten Ferret-Skripten können die Pseudovariablen _X,_Y,_Z,_T und _I,_J,_K,_L auch über ihre alten Namen Z,Y,Z,T bzw. I,J,K,L angesprochen werden. Entsprechend können nur die Variablen E,F,M und N als User-Variablen definiert werden da sie nicht als Pseudovariablen vordefiniert sind.

2.7 Subgitter (Regionen)

Bei Ferret geht der Begriff 'Region' sehr viel weiter, da alle sechs Achsen auf einen gültigen Bereich beschränkt werden können, nicht nur im Raum. Im Paragraph 2.5 wurde die Einschränkung von Achsenabschnitten schon als Variablen-Qualifier eingeführt. Dies war aber nur für die Variable gültig, für die dieses Angaben gemacht wurden. Soll die Beschränkung beispielsweise für mehrere Variablen eines arithmetischen Ausdrucks gelten, so kann der Achsenabschnitt entweder als Kommando-Qualifier gewählt werden oder über (benannte) 'regions' global festgelegt werden:

Region mit Kommando-Qualifier festlegen

Der Kommando-Qualifier zum Fetlegen der augenblicklichen Achsenbereiche lautet, ähnlich wie beim Variablen-Qualifier (nur eben mit vorangestelltem Schrägstrich):

/$\mathcal{X}$ = <START>[:<END>[:<DELTA>]]

Das $\mathcal{X}$ steht auch hier entweder für eine Achse (X,Y,Z,T,E,F) oder einen Index (I,J,K,L,M,N). Selbstverständlich können mehrere Dimensionen gleichzeitig eingeschränkt werden, indem jeweils ein Qualifier für die einzelne Achse oder Index angegeben wird. Zum Beispiel:

```
yes? list/x=20W:10E/y=0/z=0/l=1 temp,salt
```

Standard-Region festlegen

Um sich bei jedem einzelnen Kommando die Eingabe der Region zu sparen, kann vor der endgültigen Auswertung (Plot, Listing oder Abspeichern in Datei) auch eine Standard-Region definiert werden, die dann für alle nachfolgenden Kommandos und Variablen-Auswertungen gültig ist. Dies geschieht über das Kommando `set region`:

set region/$\mathcal{X}$ = <BEREICH> [**/$\mathcal{X}$ = ...**]

Die Angaben für $\mathcal{X}$ sind die gleichen wie im vorigen Abschnitt. Mit dem Kommando `cancel region` kann die Region wieder zurückgesetzt werden.

Benannte Regionen

Mit dem Kommando define region können Regionen auch benannt werden, sodass dann der Name statt numerischer Angaben im Kommando- oder Variablen-Qualifier bzw. bei der Festlegung der Standardregion genutzt werden kann:

define region/$\mathcal{X}$ = <BEREICH> [**/$\mathcal{X}$ = ...**] **<REGION-NAME>**

Ein Beispiel verdeutlicht die Verwendung der benannten Regionen:

```
yes? use levitus_climatology
yes? define region/x=100W:38E/y=30N:65N NATL
yes? define region/x=20E:130E/y=40S:30N INDIC
yes? se reg NATL
yes? shade temp[k=1,l=1]
yes? shade temp[k=1,l=1,@INDIC]
```

3 Plot-Kommandos

Es stehen eine Vielzahl von Kommandos in Ferret zur Verfügung, mit denen unterschiedliche Visualisierungen von Daten möglich sind. Einige davon lassen sich sogar kombinieren, etwa Kontur-Linien über ein Farbfeld oder Vektor-Pfeile auf einer Karte. Typische Anwendungsbeispiele sind in der folgenden Tabelle gelistet:

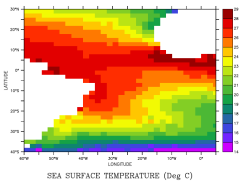
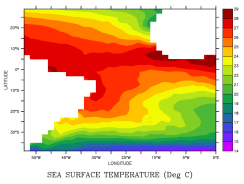
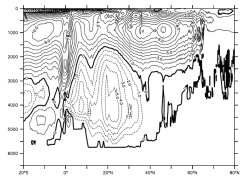
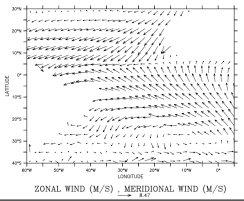
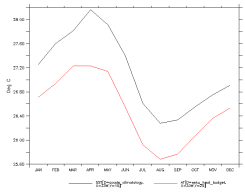
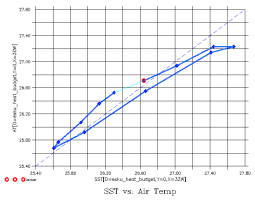
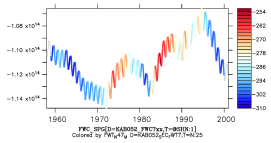
Kommando	Beispiel	Erläuterung
`shade`		2D-Flächen; Darstellung einzelner Gitterpunkte als Pixel (z.B. Karten, Schnitte, Hovmöller-Diagramme,...)
`fill`		Kombination aus `shade` und `contour`: Die Flächen zwischen den Isolinien wird entsprechend farbig ausgemalt. Randbereiche, wie Küstenlinien, können Größere Lücken aufweisen.
`contour`		2D-Iso-Linien; Boxwerte werden interpoliert für eine 'glatte' Darstellung (z.B. Stromfunktionen,...)
`vector`		Darstellung von Vektor-Pfeilen (z.B. Strömungsfelder,...)
`plot`		Einfache Linien-Diagramme (z.B. Zeitreihen, Profile,...)
`plot/vs`		Scatterplot zweier Größen gegeneinander
`ribbon`		Einfacher Linienplot, bei dem die Linie mit den Werten einer synchronen Datenreihe eingefärbt werden kann.

Tabelle 4: Übersicht der Plotbefehle.

Allgemeine Optionen

Die Plot-Kommandos haben die Form

PlotKommando*/<qualifier> *Ausdruck

Einige der Kommando-<qualifier> können bei allen Plot-Befehlen eingesetzt werden; sie sind in Tabelle 5 zusammengefasst.

Kommando-Qualifier	Erläuterung
`/nolabel`	Lässt jegliche automatische Achsenbeschriftung weg.
`/title="Mein Title"`	Zeigt den angegebenen Titel anstatt der Variablen-Namen bzw. deren Definition[9]
`/overlay`	Plottet den selben Typus (1D oder 2D) über ein existierendes Bild darüber
`/hlim=Min:Max:Delta`	Setzt die horizontale Achsbegrenzungen auf `Min`, `Max` mit optionaler Schrittweite `Delta`
`/vlim=Min:Max:Delta`	Vertikale Achsenbegrenzung wie `hlim`
`/[h|v]grat`	Zeichnet ein Gitter, entsprechend der Achseneinteilung, über die Abbildung.
`/axes=(<t>,<b>,<l>,<r>)`	Gibt an, welche Achse (mit Beschriftung) eingezeichnet werden soll (0/1)

Tabelle 5: Allgemeine Kommando-Qualifier für Plot-Kommandos.

3.1 Flächenplots (2D)

shade/<qualifier> <Ausdruck>

fill/<qualifier> <Ausdruck>

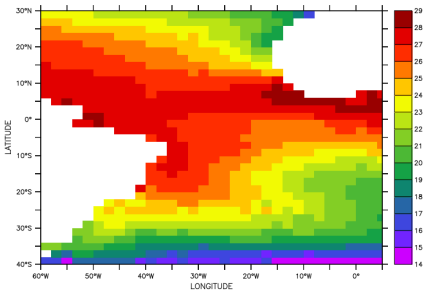
Um Flächenbilder zu erstellen, kann man die Kommandos **shade** oder **fill** (als Sonderform des Kommandos contour; siehe Kapitel 3.2) benutzen. Zum Beispiel:

```
yes? use coads_climatology
yes? se reg/x=60W:5E/y=40S:30N
yes? shade/lev=20 sst[l=12]
```

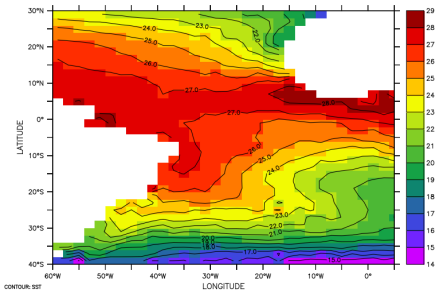
Wie in Abbildung 1a zu sehen, zeichnet shade jeden Datenpunkt als farbige Fläche. Bei groben Gittern sieht man daher großformatige Kästchen. `contour` interpoliert hingegen die Werte, sodass die Grenzlinien glatter erscheinen. Im Beispiel in Abbildung 1b werden hierzu die Kontur-Linien über ein shade mit dem folgenden Kommando gelegt:

```
yes? contour/ov/lev=20 sst[l=12]
```

[9] Unter Umständen werden die Sonderzeichen _ und ^ von Ferret zum Tief- bzw. Hochstellen des nachfolgenden Zeichens interpretiert. Um diese Verhalten abzustellen, kann der Mode ASCII_FONT eingeschaltet werden (set mode ASCII_FONT) oder die Zeichenfolge @AS vorangestellt werden (/title=@AS My_Title)



(a) Einfaches `shade`



(b) `shade` mit `contour`-Linien darüber

Abbildung 1: Horizontale Karten 1: SST im Dezember

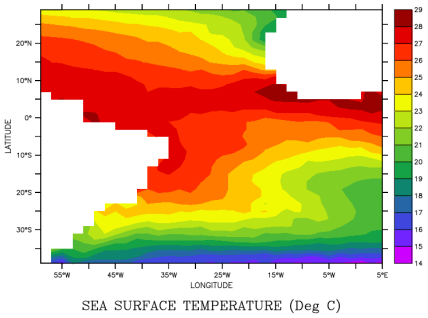
Als Alternative zum shade mit darüber gelegten Kontour-Linien, lassen sich die Bereiche zwischen den Kontour-Linien auch farblich auffüllen, mit dem Befehl fill:

```
fill/lev=20 sst[l=12]
```

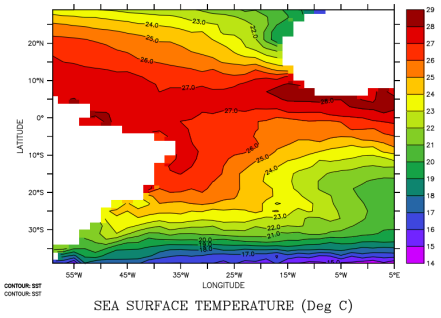
Allerdings können wegen der Interpolation für die Kontour-Linien, am Rand (Küste) zusätzliche Lücken entstehen (vgl. 2a). Mit einem Trick lassen sich diese Lücken verdecken: Zuerst wird das Feld mit einem shade-Befehl geplottet (jetzt mit dem Command Qualifier /trim) und anschließend ein fill darübergelegt. Der Qualifier /line im fill Kommando bewirkt zusätzlich das Zeichnen der Kontur-Linien und /lev ohne weitere Angaben stellt sicher, dass für fill die selben Level verwendet werden, wie für shade.

```
shade/lev=20/trim sst[l=12]
fill/lev/ov/line
```

Im letzten Beispiel wurde beim fill die Variable am Ende (`sst[l=12]`) weggelassen. Das ist kein Fehler, sondern bewirkt, dass Ferret einfach den letzten verwendeten Ausdruck (aus dem shade Befehl zuvor) nimmt.



(a) fill



(b) `shade mit fill` und `contour`-Linien.

Abbildung 2: Horizontale Karten 2: SST im Dezember

Info 6: 2D Karten-Darstellung

Manche Daten enthalten Lücken, sodass diese von den fehlenden Land-Werten nicht mehr unterschieden werden können. Zudem lassen grobe Gitter nur entsprechend ungenaue Küstenlinien zu. Wurde aber ein Etopo-Datensatz mit Ferret zusammen installiert, lassen sich Land-Masken und Küstenlinien deutlich besser darstellen. Hierzu verwednet man die Skripte `fland` und `land_detail`.

`fland`

Das Skript fland erzeugt eine Mercator-Darstellung der Weltkarte. Mit Hilfe der Argumente, die dem Skript übergeben werden lässt sich das Ergebnis beeinflussen:

```
go fland [resolution] [palette] [overlay_style] [detail] [x=lo:hi] [y=lo:hi]
```

Argument	Erläuterung
`resolution`	may be 120,60,40,20,10, or 5 default=20. To use resolution "nnn" the file etoponnn must exist.
`palette`	may be any palette on your system. Normally solid colors, like black, gray,red, etc. (see Fpalette '*') default=gray
`overlay_style`	"basemap", "transparent", or "overlay" (default)
`detailed`	may be "detailed", "all" or "solid" default=solid. Use "detailed" together with a spectrum like dark_terrestrial Use "all" to plot oceans as well as land.
`longitude limits`	specify as for any FERRET longitude range
`latitude limits`	specify as for any FERRET latitude range

`land_detail`

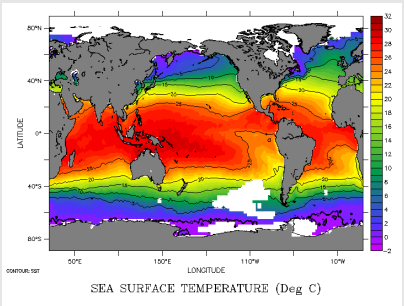
Um nur die Küstenlinien zu zeichnen kann man land_detail verwenden (kann aber auch Flussläufe und Landes-Grenzen):

```
go land_detail [1:continent] [2:basemap_style] [3:country] [4:state] [5:rivers]  \
                        [6:more_rivers] [7:marine_boundaries]
```

Argument	Erläuterung
`1:continent, 3:country, 4:state, 5:rivers, 6:more_rivers, 7:marine_boundaries`	Line style (or Pen), e.g. 1,2,3,...7,8...
`2:basemap_style`	"basemap" "overlay", or omitted for overlay

Beispiel:

```
yes? fill sst[l=1]
yes? contour/ov sst[l=1]
yes? go fland 20
yes? go land_detail
```



Level

Die Anzahl der Level (Kontour-Linien bzw. Anzahl der unterschiedlichen Farbflächen) lässt sich mittels Qulaifier **/level=** festlegen:

Angabe (/level=)	Beispiel	Erläuterung
#	10	Anzahl der Level (Minimum, Maximum und Delta werden automatisch gesetzt)
#d	2d	Delta=2 (Min., Max, und Anzahl werden automatisch gesetzt)
c	c	Zentrierte Level (Min, Max, Delta, Anzahl automatisch)
#c	10c	Anzahl der zentrierte Level (Min, Max, Delta automatisch)
(#)	(0)	Ein bestimmtes Level
(<min>,<max>,<delta>)	(-10,10,1)	Minimum, Maximum und Delta gegeben (Anzahl automatisch)
(-inf)	(-inf)	Ein Level für minus Unendlich
(+inf)	(+inf)	Ein Level für plus Unendlich
(<lev-1>)(<lev-2>)(...)	(-inf)(-10,0,1)(0,5,.5)(inf) Die geklammerten Level-Angaben lassen sich für eine nicht-lineare Farbskala kombinieren	

Tabelle 6: Kommando-Qualifier /level

Farbpalette

Für shade bzw. fill nimmt Ferret immer erstmal die aktuelle Default-Farbpalette. Es steht jedoch eine Reihe weiterer Farbpaletten zur Verfügung. Der UNIX-Befehl

```
> Fpalette '*'
```

oder innerhalb von Ferret:

```
yes? spawn Fpalette '*'
```

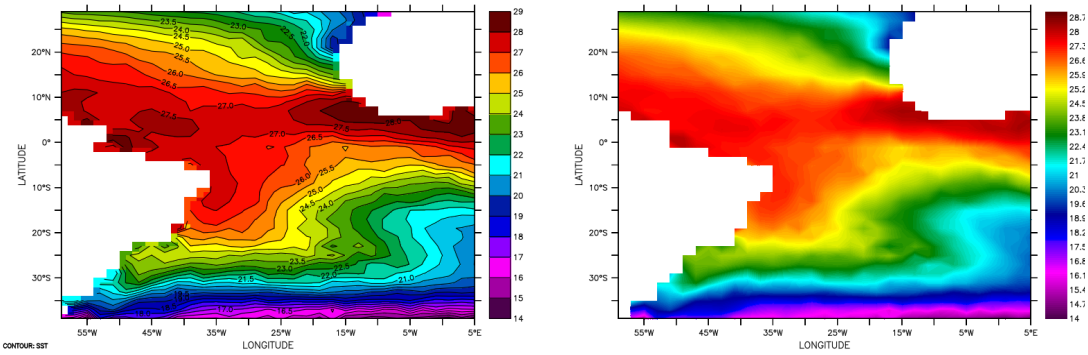
erzeugt eine Liste der Paletten[10], die von Ferret gefunden werden und vom Benutzer über die Option **/palette=** dann genutzt werden können, z.B. für Abbildung 3a:

```
yes? shade/pal=broad/lev=20 sst[l=12]
yes? fill/ov/lev/line
```

Manchmal ist es wünschenswert, besonders weiche Farbübergänge zu haben. Für diesen Fall sollte das Delta in der /level-Angabe möglichst klein oder die Gesamtzahl an Farblevel möglichst hoch gewählt werden (Gesamtzahl der Farben sollte aber nicht viel mehr als 100, in wenigen Fällen evt. 200, betragen, da die Dateigröße des Bildes sonst nur unnötig vergrößert wird). Zusätzlich kann ein Farbbalken vom Typ 'contiuous' genutzt werden (/key=cont), wie in Abbildung 3b gezeigt:

```
yes? fill/key=cont/lev=(14,29,.1) sst[l=12]
```

[10] Siehe auch Anhang Anhang B Farb-Paletten



(a) shade+fill+contour-Linien mit andere Farbskala

(b) fill mit >100 Farben.

Abbildung 3: fill Varianten mit anderer Farbskala

3.2 Konturplots (2D)

contour/<qualifier> <Ausdruck>

Bei den Flächenplots wurden Konturlinien schon verwendet, um die Grenzen zwischen den einzelnen Farben besser zu unterscheiden. Für manche Größen (z.B. eine Stromfunktion) ist der Verlauf der Konturlinien ausschlaggebend (vgl Abbildung 4a):

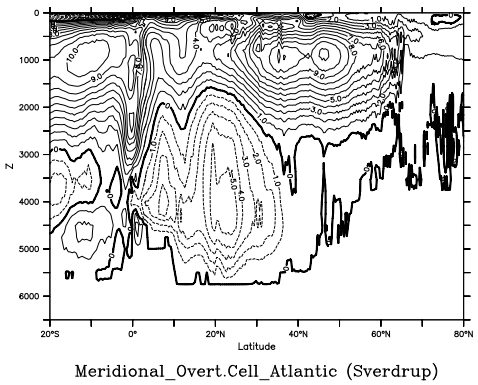
```
yes? use Atlantic_moc.nc
yes? contour/nolab/lev=(-10,20,1)/vlim=6500:0:-500/hlim=-20:80 \
         ZOMSFATL,nav_lat,z[g=zomsfatl]
```

Um bestimmte Bereiche hervozuheben, können diese mit einer Farbe hinterlegt werden. Hierzu wird ein passender fill Befehl hinter die Kontour-Linien gelegt. Im Beispiel 4 (Abbildung 4b) wurde die antizyklonale (im Uhrzeigersinn drehende; positive Werte) Zirkulation mit einem hellen Grau hinterlegt und der Farbbalken weggelassen. Dadurch rückt die Bodenzelle in Weiß stärker in den Vordergrund. Zusätzlich wurden die +2 und -2 Konturen farblich hervorgehoben (siehe hierzu den nächsten Abschnitt “Linienfarben”).

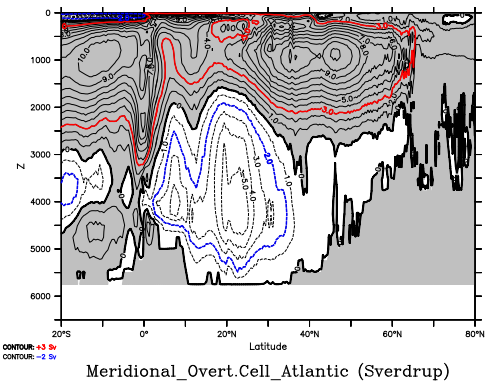
Beispiel 4

```
yes? use Atlantic_moc.nc
yes? contour/nolab/lev=(-10,20,1)/vlim=6500:0:-500/hlim=-20:80 \
        ZOMSFATL,nav_lat,z[g=zomsfatl]
yes? cancel mode ASCII
yes? fill/pal=grey_light/lev=(0,30)/nokey/vlim=6500:0:-500/hlim=-20:80 \
       /title="Mittlere AMOC (@C002+3 Sv @C001,@C004-2 Sv@C001)" \
       ZOMSFATL,nav_lat,z[g=zomsfatl]
yes? contour/ov/nolab/lev=(-10,20,1)/vlim=6500:0:-500/hlim=-20:80

yes? contour/ov/nolab/lev=(+3)/color=2/sigdig=1/thick=2
yes? contour/ov/nolab/lev=(-2)/color=4/sigdig=1/thick=2
```



(a) einfacher contour plot.



(b) Wie (a) aber mit grau hinterlegten positiven Werten und zwei farblich betonten Isolinien (-2/+2Sv).

Abbildung 4: contour Varianten der meridionalen Umwälzzelle im Nordatlantik.

Linienfarben

Um die Farbe der Isolinien bei Contour zu verändern, kann der Kommando-Qualifier /color genutzt werden. Für die Farb-Angabe stehen drei Varianten zur Verfügung:

- `/color = <FarbNummer>`
- `/color = <FarbName>`
- `/color = (<R>,<G>,<B>)`

Es gibt eine in Ferret voreingestellte Farbreihenfolge (siehe Tabelle 7). `<FarbNummer>` entspricht dabei dem Index dieser Reihenfolge. Statt der Nummer kann auch der entsprechende `<FarbName>` gewählt werden; die Strichstärke kann dann mit dem Qualifier `/thickness=` beeinflusst werden. Zuletzt kann die Farbe auch mit einem RGB Farbwerttripel $(0 - 100\%)$ angegeben werden und ist die flexibelste Variante.

Folgende drei `Contour`-Kommandos führen zum gleichen Ergebnis: die Konturlinien werden blau gezeichnet:

```
yes? contour/color=4 sst
yes? contour/color=blue
yes? contour/color=(0,0,100)
```

Textfarbe

Ähnlich wie die Linienfarben können auch die Farben von Label und Titel in einer Grafik beeinflusst werden. Allerdings steht hier nur die Farbtabelle vordefinierter Farben zur Verfügung (Tabelle 7). Um die Farbe in einem Textstring zu ändern wird der folgende Code an der Stelle eingefügt, ab der die Frabe wirksam werden soll:

`" @C`**`nnn`**` "`

Dabei steht **nnn** für die Farbnummer (vgl. Tabelle 7) als ganze, dreistellige Zahl mit führenden Nullen. Im Beispiel 4 wird auf diese Weise im Titel die Textteile “-2 Sv” und “+3 Sv” in blau (`@C004`) bzw. rot (`@C002`) ausgegeben.

Farb-Nr.	Farbe	RGB Code
1	**black**	(0,0,0)
2	**red**	(100,0,0)
3	**green**	(0,100,0)
4	**blue**	(0,0,100)
5	**lightblue**	(0,100,100)
6	**purple**	(100,0,100)
Mit der 7. Farbe wiederholen sich die Farben, allerdings mit einer dickeren Strichstärke. (Standardmäßig kennt Ferret also 6 Farben. Das kann über `SET MODE LINECOLORS` *geändert werden.)*		
7-12	*doppelte Strichstärke*	`/col=(rrr,ggg,bbb)/thick=2`
13-18	*dreifache Strichstärke*	`/col=(rrr,ggg,bbb)/thick=3`
19	white	(100,100,100)

Tabelle 7: Vordefinierte Farben.

3.3 Vektorplot (2D)

vector/<qualifier> <U-Ausdruck>,<V-Ausdruck>

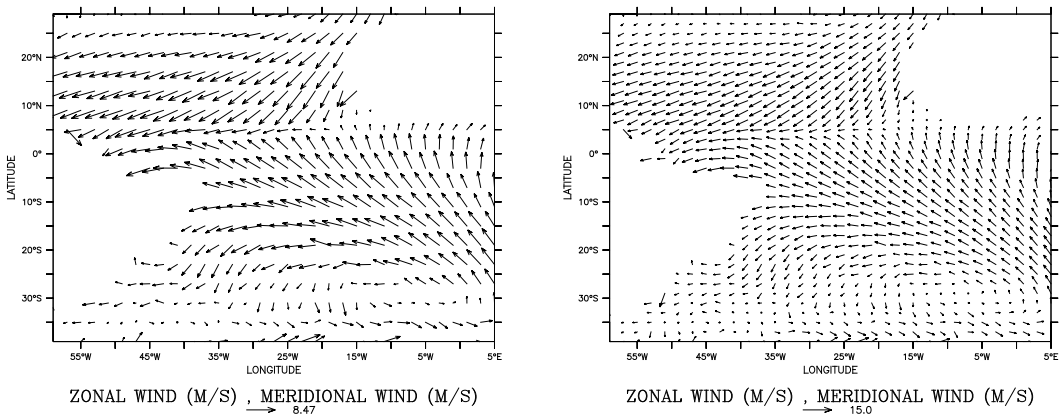
Zur Darstellung von Geschwindigkeiten, etwa Wind oder Ozeanströmungen, können Vektorplots genutzt werden. Für jeden Gitterpunkt ist eine U- und eine V-Komponente nötig. Da Ferret versucht, selbst schon die optimale Darstellung herauszufinden, ist das einfache vector-Kommando immer ein guter Einstiegspunkt (Abbildung 5a), z.B.:

```
yes? use coads_climatology
yes? se reg/x=60W:5E/y=40S:30N/l=12
yes? vec UWND,vWND
 Using every   2th vector in the Y direction
```

Die Meldung “Using every 2th vector in the Y direction” besagt, dass Ferret nur jeden zweiten Gitterpunkt genommen hat, damit die Pfeile sich nicht zu stark überlagern. Unterm Titel wird zudem die Skalierung der Pfeile angegeben: Der Normpfeil im Beispiel hat eine Länge von 8.47.

Der Benutzer kann die Darstellung natürlich selbst beeinflussen, etwa wenn jeder Gitterpunkt dargestellt werden soll. Mittels /xskip= und yskip= wird bestimmt, welcher Gitterpunkt in X- und Y-Richtung berücksichtigt werden soll (1 steht für jeden Punkt). Darüber hinaus lässt sich mit /length die Skalierung des Norm-Pfeiles bestimmen (siehe Abbildung 5b):

```
yes? vec/xskip=1/yskip=1/len=15 UWND,VWND
```



(a) Mit automatischer Skalierung.

(b) Mit bestimmter Vektor-Zahl und -Länge.

Abbildung 5: vector-Plot

Leicht lässt sich ein Vektor-Plot auch über einen anderen Flächenplot legen. Als Beispiel soll hier die spezifische Luftfeuchte (Variable SPEH im coads-Datensatz) im Hintergrund mit Farbskala '`yellow_green_blue`' geplottet, darüber die Vektor-Pfeile und die Pfeile mit einer Absolutgeschwindigkeit größer 6 m/s rot hervorgehoben (diese Technik wird im Kapitel 4.4) erklärt):

```
yes? use coads_climatology
yes? se reg/x=60W:5E/y=40S:30N/l=12
yes? let SPD=(UWND^2+VWND^2)^.5
yes? fill/pal=yellow_green_blue/lev=(5,40,.5)) SPD
yes? vec/ov/xskip=1/yskip=1/len=12/col=1 UWND,VWND
yes? vec/ov/xskip=1/yskip=1/len=12/col=red/thick=2 \
       UWND*ignore0(SPD gt 6),VWND*ignore0(SPD gt 6)
yes? go fland
```

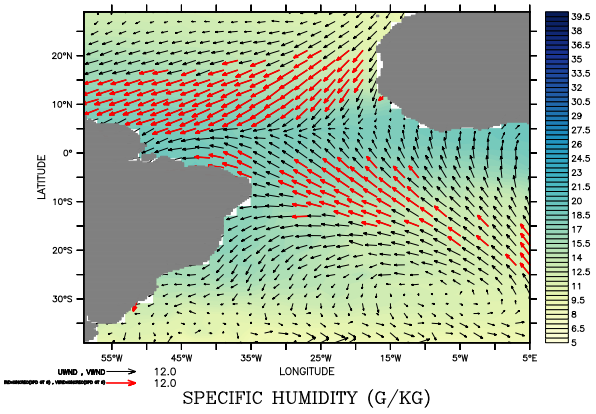


Abbildung 6: Vektor-Plot der COADS-Windgeschwindigkeit im Dezember (absolute Geschwindigkeiten $>$ 6m/s in Rot) mit spezifischer Luftfeuchte im Hintergrund.

3.4 Linien-/Punkt-Plots (1D)

plot/<qualifier> <Ausdruck>

Einfache Linien-Plots lassen sich mittels **plot** zeichnen: Eine Zeireihe der SST an einem Gitterpunkt (am Äquator und 32°W) zum Beispiel, lässt sich wie folgt als Linienplot darstellen:

```
yes? use coads_climatology
yes? plot sst[y=0,x=32W]
```

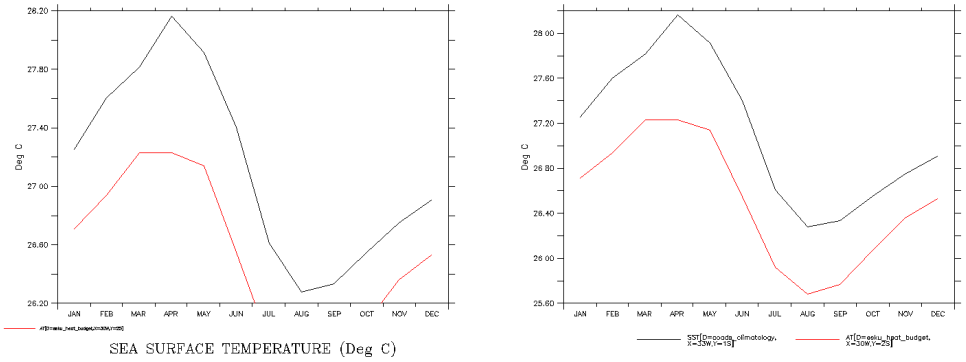
Will man dann z.B. einen anderen Datensatz darüberplotten (Abb. 7a), nutzt man den **/overlay** Schalter (vgl. Tabelle 8):

```
yes? use esku_heat_budget
yes? plot/ov at[y=0,x=32W]
```

Hier wird der neue default-Datensatz 'esku_heat_budget' gleich als neuer default verwendet. Genauer wäre es, die Spezifikation `d=2` mit anzugeben:

```
yes? plot/ov at[y=0,x=32W,d=2]
```

Die Farbe für die übereinander liegenden Linien wählt Ferret automatisch, sofern der Benutzer nichts anderes einstellt (Option /color=; siehe auch Tabellen 7 und 8) . Die Farb-Reihenfolge folgt der gleichen wie beim contour-Plot.



(a) Plot mit Overlay

(b) Zwei Größen (Datensätze) gleichzeitig

Abbildung 7: Linien-Plot: SST aus zwei Datensätzen

In Abbildung 7a liegen die Werte des zweiten Datensatzes (rot) leider außerhalb der vertikalen Plot-Achse, da Ferret die vorhandene Achse nicht nachträglich verändern kann. Das lässt sich beheben, indem man beide Größen gleichzeitig plottet.
ACHTUNG: Da der aktuelle Datensatz der zuletzt eingelesene Zweite ist, müssen wir für die SST den Qualifier `[d=1]` hinzufügen:

```
plot sst[d=1,y=0,x=32W],at[d=2,y=0,x=32W]
```

Eine andere Möglichkeit besteht darin, die Achsen-Grenzen beim ersten Plot-Befehl per Hand mit der Option **/vlim=** festzulegen (vgl. Tabelle 8).

Kommando-Qualifier	Erläuterung
/col=#	Farb-Nummer #
/thickness=#	Strichstärke (1-3)
/vlim=25.4:27.8:.2	Begrenzung und Intervall der Ordinate (Y)
/hlim=25.4:27.8:.2	Begrenzung und Intervall der Abzisse (X)
/tr	Transpose: vertauscht X- und Y-Achse
/vs	Zwei Größen gegeneinander plotten
/line	Verbinde die Punkte mit einer Linie
/sym=22	Verwende Symbol 20 zum plotten der Datenpunkte (siehe Abbildung 9)
/grat	Zeichne Gitterlinien (auch: /hgrat bzw. vgrat)

Tabelle 8: Kommando-Qualifier für Kommando plot.

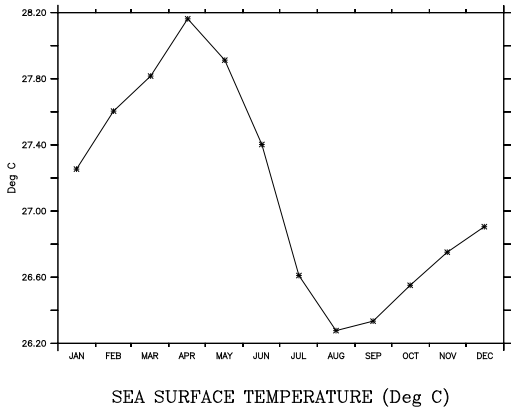


Abbildung 8: Linienplot mit Symbol 22 zum markieren der Datenpunkte.

Punktplot

Statt eines Linienplots können auch die einzelnen Werte (zusätzlich oder allein) mit Symbolen (siehe Abbildung 9) in das Diagramm eingetragen werden:

```
plot/sym=22/line sst[d=1,y=0,x=32W]
```

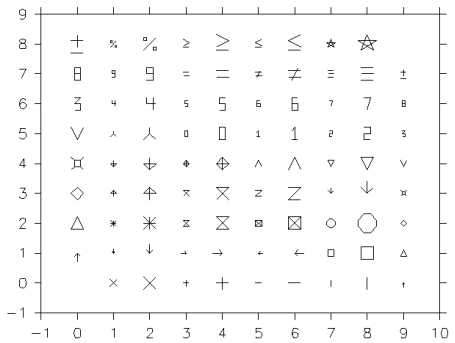


Abbildung 9: Plot-Symbole. Den Symbol-Wert für /sym= erhält man aus: Y-Achsenwert*10 + X-Achsenwert. Der Wert 35 entspricht z.B. “z” während 36 ein großes “Z” darstellt.

Hysterese-Plot

Desweiteren lassen sich auch zwei Variablen mittels **plot/vs** gegeneinader plotten, z.B. für eine Hysterese. Um, zum Beispiel, die SST am Äquaor gegen die Luft-Temperatur an gleicher Stelle zu plotten (beide aus dem esku-Datensatz, siehe Abbildung 10), definiert man zwei Variablen (`sst_eq` und `sat_eq`) und plottet die Werte der ersten Variable auf der X-Achse gegen die Werte der zweiten Variablen auf der Y-Achse (beide Variablen müssen natürlich die selbe Zahl von Elementen haben):

```
yes? let sst_eq=sst[d=2,y=0,x=30W]
yes? let sat_eq=at[d=2,y=0,x=30W]
yes? plot/vs/line/sym=30/col=blue/thick=3/hlim=25.4:27.8:.2\
       /vlim=25.4:27.8:.2/grat/title="SST vs. Air Temp" sst_eq,sat_eq
yes? plot/ov/vs/col=blue/thick=1/dash/nolab {25.4,27.8},{25.4,27.8}
yes? plot/ov/col=2/thick=3/sym=28/vs/title="Januar"/L=1 sst_eq,sat_eq
```

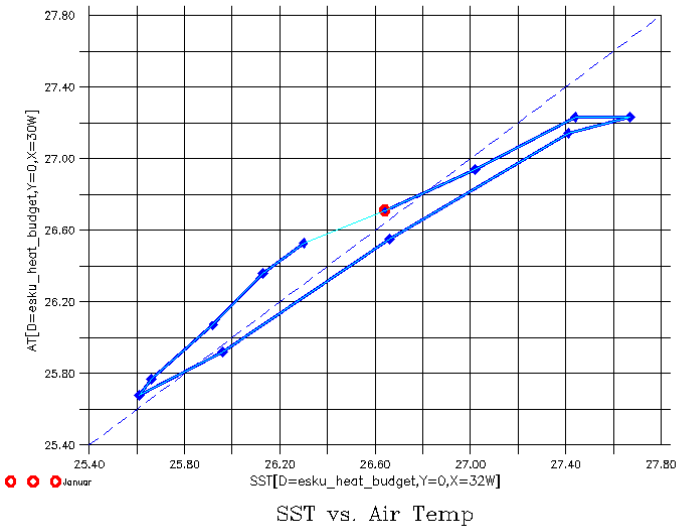


Abbildung 10: Linien-Plot: SST vs. Luft Temperatur

Die Datenpunkte werden dabei als Rauten dargestellt (/sym=30; vgl Abbildung 9) und mit einer blauen mittel-dicken Linie verbunden. Darüber ist die Diagonale gleicher Luft- und Meeresoberflächen-Temperatur als dünne gestrichelte, blaue Linie gelegt (Das Label wurde mit **/nolab** weggelassen). Der rote Kreis markiert den Januar-Wert (L=1). Der Backslash am Ende der 3. Zeile oben (`.../hlim=25.4:27.8:.2\`) erlaubt es, ein langes Kommando in der nächsten Zeile fortzuführen. Dies ist jedoch nur eine Darstellungsweise für dieses Skript; in Ferret kann die Zeile einfach ohne Backslash in einer Zeile fortgeschrieben werden.

Für eine geschlossene Hysterese-Kurve muss man explizit 13 Monate mit der Option **/L=1:13** mit angeben, damit der 12. mit dem 13. Wert (entspricht wieder dem ersten, da sich wiederholende Modulo-Zeit-Achse) mit einer Plot-Linie Verbunden wird (als dünne hellblaue Linie):

```
yes? plot/ov/col=2/line=5/vs/nolab/L=1:13 sst_eq,sat_eq
```

Ribbon

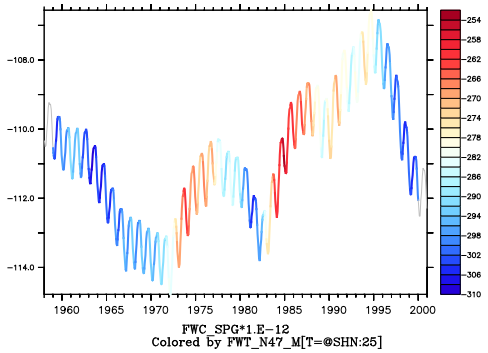
Eine besondere Variante des Linienplots ist der farbige Bänder-Plot (auch Ribbon genannt): Dabei zeigt die Kurve über die Y-Achse die Werte einer Variablen an, während der Linien-Verlauf mithilfe einer zweiten Variablen eingefärbt wird (ähnlich wie bei den Flächenplots stehen hier sämtliche Ferret-Farbskalen zur Verfügung).

Abbildung 11a zeigt als Beispiel den zeitlichen Verlauf des Süßwassergehalts im subpolaren Nordatlantik (`FWC_SPG`; linke Y-Achse), der mit dem Süßwassertransport über 47°N (`FWT_47N_M`) hinweg eingefärbt wurde. Letzere Variable wurde noch mittels Achsentransformation Tiefpass-gefiltert (Siehe hierzu Kapitel 4.2).

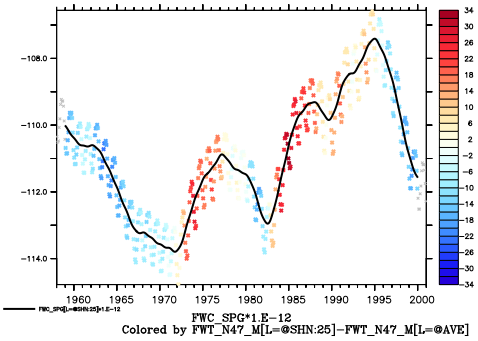
```
yes? use freshwater.nc
yes? plot/ribbon/thick=3/pal=blue_darkred \
      FWC_SPG*1.e-12,FWT_N47_M[l=@SHN:25]
```

Um die langperiodischen Schwankungen des Süßwassergehaltes zu betonen, kann diese als durchgängige schwarze Linie dargestellt werden, während die monatlichen FWC-Werte als farbige Punkte mit entsprechendem FWT-Farbwert in den Hintergrund treten (Abbildung 11b):

```
yes? plot/ribbon/thick=3/pal=blue_darkred/sym/lev=c \
      FWC_SPG*1.e-12,FWT_N47_M[l=@SHN:25]-FWT_N47_M[l=@AVE]
yes? plot/thick=3/col=black/line/ov FWC_SPG[l=@SHN:25]*1.e-12
```



(a) Zum Einfärben der FWC-Zeitreihe wurde die monatliche FWT-Zeitreihe gefiltert, sodass deren zwischenjährliche und dekadische Schwankungen hervortreten.



(b) Statt eines Linienplots sind hier die einzelnen Werte der FWC-Zeitreihe als Punkte dargestellt. Die Färbung kommt von den langperiodischen Anomalien der FWT-Zeitreihe.

Abbildung 11: Verschiedene Varianten eines plot/ribbon-Plots: Süßwasseranteil eingefärbt mit Süßwassertransporten über 47°N hinweg.

4 Analysis

4.1 Arithmetische Operationen

Im Abschnitt 2.3 haben wir bereits gesehen, wie einfache Variablen definiert werden können, nämlich mit `let <Variable> = <Ausdruck>`. Für den Ausdruck auf der rechten Seite stehen die typischen Operatoren (Tabelle 9) zur Verfügung und es gelten die bekannten Rechenregeln für Multiplikation und Addition und Klammerung.

Operator	Erläuterung
+	Addition
-	Substraktion
*	Multiplikation
/	Division
^	Exponent
( )	Klammer

Tabelle 9: Arithmetische Operatoren

Beispele:

```
yes? let a = 2
yes? let b = a + 6
yes? let c = b / a
yes? let d = c ^ .5
```

Das Ergebnis kann dann mit `list` ausgegeben werden (siehe auch Kapitel 7.1):

```
yes? list a,b,c,d
 Column  1: A is 2
 Column  2: B is A + 6
 Column  3: C is B / A
 Column  4: D is C ^ .5
              A      B      C      D
I / *:     2.000  8.000  4.000  2.000
```

Bei vielen Variablen-Abhängigkeiten kann schnell der Überblick verloren gehen. Hier hilft das Kommando `show variable/tree VAR`, das alle Variablendefinitionen (entsprechend eingerückt) auflistet, die zu der angegebenen Variablen führen:

```
yes? sh va/tree d
   D = C ^ .5
     C = B / A
       B = A + 6
         A = 2
       A =  (defined above)
```

Eingebette Auswertung

Für skalare Größen kann auch die eingebette Auswertung genutzt werden. Dabei wird der Ausdruck zwischen den rückgestellten Anführungsstrichen (` `) sofort ausgewertet und durch das Ergebnis ersetzt.

```
yes? let e = `a` + `2 * c`
 !-> DEFINE VARIABLE e = 2 + 8

yes? sh va e
 E = 2 + 8
```

Zudem kann man das Zahlenformat der eingebetteten Auswertung beeinflussen und sogar einige Charakteristika einer Variablen bzw. eines Ausdrucks in Erfahrung bringen. Hierzu werden hinter dem Ausdruck, mit Komma abgetrennt, verschiedene Schlüsselwörter aufgelistet.

`*Ausruck*,<Schlüssel>`

Für <Schlüssel> gibt es eine Reihe von Schlüsselwörtern (nur der unterstrichene Anteil muss angegeben werden):

PRECISION=<n>	: Anzahl der Stellen
ZW=<n>	: Anzahl der Vorkomma-Stellen inkl. führender Nullen
BAD=<bad>	: Ersatzwert, wenn vorangestellter Ausdruck ungültig ist
RETURN [= <RVAL>]	: Abfrage diverser Eigenschaften des Ausdrucks

PRECISION=<n> Ist <n> positiv dann werden insgesamt <n> Vor- und Nachkommastellen ausgegeben. Ein negatives <n> gibt die Anzahl nur der Nachkommastellen an (Bei ganzen Zahlen wird die Null nach dem Komma grunsätzlich abgeschnitten).

ZW In manchen Fällen müssen Zahlen mit führenden Nullen ausgegeben werden. Dies kann mit dem Schlüsselwort ZW erreicht werden.

BAD Im Falle eines ungültigen Ausdrucks kann mit BAD der zurückzugebende Fehlerwert bestimmt werden.

RETURN Anders als mit den vorangegenagenen Schlüsselwörtern kann mit RETURN nicht das Format der Ausgabe beeinflusst werden, sondern es können Eigenschaften des Ausdrucks abgefragt werden:

RETURN=	**Erläuterung**
size	Gesamtzahl an Datenpunkten
istart, jstart, kstart, lstart, mstart, nstart	Start-Index entlang der entsprechenden Achse
iend, jend, kend, lend, mend, nend	End-Index entlang der entsprechenden Achse
xstart, ystart, zstart, tstart, estart, fstart	Start-Achsenwert entlang der entsprechenden Achse
xend, yend, zend, tend, eend, fend	End-Achsenwert entlang der entsprechenden Achse
isize, jsize, ksize, lsize, msize, lsize	Anzahl der Datenpunkte entlang einer bestimmten Achse
bad	Ersatzwert für einen Missing Value (NaN)
calendar	Typ des Kalenders
T0	Referenz-Zeitpunkt
units	Physikalische Einheit des Ausdrucks
iunits, junits, kunits, lunits, munits, nunits	Physikalische Einheit der entsprechenden Achse
xunits, yunits, zunits, tunits, eunits, funits	Physikalische Einheit der entsprechenden Achse
title	Langer Name der Variable
grid	Name des verknüpften Gitters
iaxis, jaxis, kaxis, laxis, maxis, naxis	Name der entsprechenden Achse
xaxis, yaxis, zaxis, taxis, eaxis, faxis	Name der entsprechenden Achse

RETURN=	Erläuterung
dset, dsetnum, dsetpath, dsettitle	Name, Nummer, absoluter Pfad und Titel des verknüpften Datensatzes
nc_scale, nc_off	NetCDF Atrribute
user_scale, user_off	Benutzerdefinierte Attribute
dtype	NetCDF Datentyp
xmod	Länge einer X-Modulo Achse
tmod	Länge einer T-Modulo Achse
isDepth	Flag für Tiefenachse (1: ja; 'normal':nein)

4.2 Achsen-Transformationen

Manche Opertionen sind an eine Achse gebunden, wie etwa ein räumliches Integral oder ein zeitliches Mittel. Für solche Fälle bieten sich sog. Achsen-Transformationen an, die als Variablen-Qualifier angegeben werden:

$$\textbf{\textit{variable}}[\ \mathcal{X}\ \texttt{=}\ \texttt{@<TRA>}\ ]$$

Dabei steht $\mathcal{X}$ entweder für eine Achse (X,Y,Z,T,E,F) oder einen Index (I,J,K,L,M,N). Die entsprechende Operation @<TRA> (siehe Tabelle 11) wird nur in der Dimension ausgeführt, für die die Transformation auch angegeben wird; die restlichen Dimensionen bleiben unverändert.

Transformation @<TRA>	Erläuterung
@AVE	Mittelwert
@MIN	Minimum
@MAX	Maximum
@VAR	Varianz (Standardabweichung=$\sqrt{VARIANZ}$)
@SUM	Summe
@RSUM	Laufende Summe
@DDC	Zentrierte Ableitung
@DIN	Integral
@IIN	Fortlaufendes Integral
@SHN:n	Hanning-Filter (gewichtet; Boxbreite $n \in \mathbb{N}_{\text{ungerade}}$)
@SBX:n	Rechteckiger Box-Filter (ungewichtet; Boxbreite n)
@NGD	Anzahl von gültigen Werten (ohne missing values)

Tabelle 11: Achsen-Transformationen.

Um beispielsweise die zeitliche Varianz der zonal gemittelten monatlichen Oberflächentemperatur aus COADS zu plotten, genügt folgende Eingabe[11]:

```
yes? use coads_climatology
yes? plot/tr sst[x=@AVE,t=@VAR]
```

[11] Der Kommando-Qulaifier /tr wird hier verwendet, um die geographische Breite auf die vertikale Achse abzubilden.

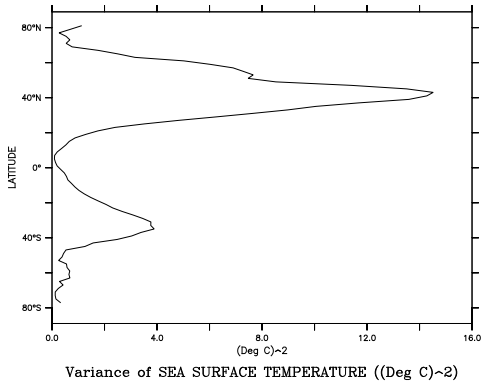


Abbildung 12: Varianz der zonal gemittelten monatlichen SST.

4.3 Funktionen

Natürlich stehen auch verschiedene Funktionen zur Verfügung. Eine Liste aller Funktionen lässt sich mittels

```
yes? show function
```

erzeugen (Siehe auch Anhang Anhang E). Eine Übersicht der wichtigsten arithmetischen Funktionen zeigt Tabelle 12.

Funktion	Erläuterung
`EXP(X)`	e-Funktion (e^{X})
`LOG(X)`	Logarithmus zur Basis 10
`LN(X)`	natürlicher Logarithmus zur Basis e
`ABS(X)`	Absolutwert
`MOD(A,B)`	Modulo oder Rest von A durch B
`INT(X)`	Ganzzahliger Anteil (abgerundet)
`SIN(`θ`), COS(`θ`), TAN(`θ`)`	Hyperbolische Funktionen (θ in Radiant: $1rad = {}^{180°}/_{\pi}$)

Tabelle 12: Arithmetische Funktionen in Ferret.

Daten-Manipulation

Missing Values Elemente eines Vektors oder einer Matrix, die keinen spezifischen Wert haben oder deren Variablen-Definition kein gültiges Ergebnis liefert, erhalten in Ferret den Ersatz-Wert "Missing Value" (in Matlab häufig NaN). In graphischen Darstellungen werden sie nicht ausgegeben und in manchen arithmetischen Operationen nicht berücksichtigt. Mit der Funktion `MISSING(A,B)` können missing values in der Variablen A durch den Wert B ersetzt werden während umgekehrt `IGNORE0(X)` überall in X den Wert 0.0 (Null) durch missing valus ersetzt. Mit `COMPRESSI` (bzw. für die anderen Dimensionen entsprechend `COMPRESS[J|K|L|M|N]`) wiederum werden alle Daten in der I- (JKLMN-) Richtung so verdichtet und neu sortiert, dass alle missing values an das Ende der I- (JKLMN-) Achse verschoben werden.

Funktion	Erläuterung
IGNORE0(X)	Ersetzt Werte 0.0 in X durch Missing Value (NaN)
MISSING(A,B)	Ersetzt Missing Values in A durch Wert B
COMPRESSI(X)	Verschiebt alle Missing Value Einträge an das Ende der Achse Auch als COMPRESS[J|K|L|M|N](X)
XSEQUENCE(VAR)	Erzeugt aus Punkt-Daten in VAR eine Variable entlang der X-Achse Auch als [Y|Z|T|E|F]SEQUENCE(VAR)
XREVERSE(VAR)	Kehrt die X-Achsenrichtung um. Auch als [Y|Z|T|E|F]REVERSE(VAR)
SAMPLEI(TO_BE_SAMPLED,X_INDICES)	Sortiert die Einträge in TO_BE_SAMPLED entsprechend der Indizes X_INDICES neu. Auch als SAMPLE[J|K|L|M|N](A,B)
MIN(X)	Globaler Minimum-Wert von X
MAX(X)	Globaler Maximum-Wert von X

Tabelle 13: Funktionen zur Datenmanipulation.

Tupel erzeugen Wie noch später im Kapitel 7.2 gezeigt wird, kann der Benutzer auch selbst Daten-Tupel[12] eingeben, sogenannte Punkt-Daten. Um diese an einer bestimmten Achse auszurichten, gibt es die Funktion XSEQUENCE() bzw. entsprechend den übrigen Achsen: [Y|Z|T|E|F]SEQUENCE().

Permutieren In manchen Fällen müssen Serien von Datenpunkten neu und in einer bestimmten Weise angeordnet werden. Ferret erlaubt das Umsortieren in einer Dimension anhand eines Index-Tupel als Argument in der Funktion z.B. für die X-Achse: SAMPLEI() (Funktion SAMPLE[JLMN]() für die übrigen Dimensionen).

Globale Extrema Um ein Minimum (oder Maximum) global in allen Dimensionen zu suchen, gibt es die Funktion MIN() bzw. MAX() statt mehrerer Variablen-Qualifier:
VAR[x=@MIN,y=@MIN,z=@MIN,t=@MIN,e=@MIN,f=@MIN].

Spezielle Funktionen

Für spezielle Aufgaben bietet Ferret einige zusätzliche Funktionen, wie etwa zur Berechnung der Meerwasserdichte oder der Zahl der Tage seit dem 1.1.1900 sowie zur Bestimmung statistischer Größen (siehe Tabelle 14).

Funktion	Erläuterung
RHO_UN(salt,temp,pref)	UNESCO density
DAYS1900(year,month,day)	Anzahl der Tage seit 1. Januar 1900
THETA_FO(salt,temp,p,ref)	Potentielle Temperatur nach Fofonoff (1977)

Tabelle 14: Spezielle Funktionen in Ferret.

[12] Ein (n-)Tupel ist eine geordnete Liste von endlich vielen Elementen.

4.4 Masken

Eine Stärke von Ferret ist das Maskieren von Daten anhand simpler Kriterien. Dabei gibt es zwei Formen: Inline-Masking und Variablen-Masking. Zwei Beispiele sollen ihre Funktionsweise erklären:

Inline-Masking

Die Bodentopographie des Ozeans soll als shade-Grafik dargestellt werden, während das Land ausgeblendet werden soll. Als Beispiel dient die 120-Minuten Version des ETOPO-Datensatzes. Zunächst wird der komplette Datensatz geplottet (Abbildung 14a). Als Farbpalette dient "land_sea" und wir legen die Farb-Level selbst fest (zwischen +/- 6500m in 500m Schritten).

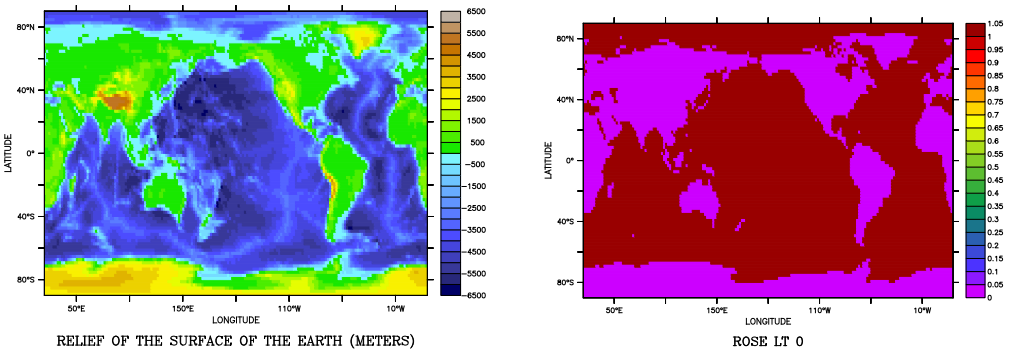
```
yes? use etopo120
yes? ! (a)
yes? shade/pal=land_sea/lev=(-6500,6500,500) rose
```

Im zweiten Schritt soll eine Maske erstellt werden, die den Wert 1 im Ozean hat und 0 an Land (14b). Dazu wird hinter die Variable **`rose`** die Bedingung "kleiner als Null: **`lt 0`**" notiert (mögliche Operatoren sind in Tabelle 13 aufgeführt):

```
yes? ! (b)
yes? shade rose lt 0
```

Operator	Erläuterung
lt	Kleiner als (less than; <)
le	Kleiner gleich (less equal; <=)
eq	Gleich (equal; =)
ge	Größer gleich (greater equal; >=)
gt	Größer als (greater than; >)
ne	Nicht gleich (not equal; !=)

Abbildung 13: Vergleichsoperatoren



(a) Vollständige Topographie.

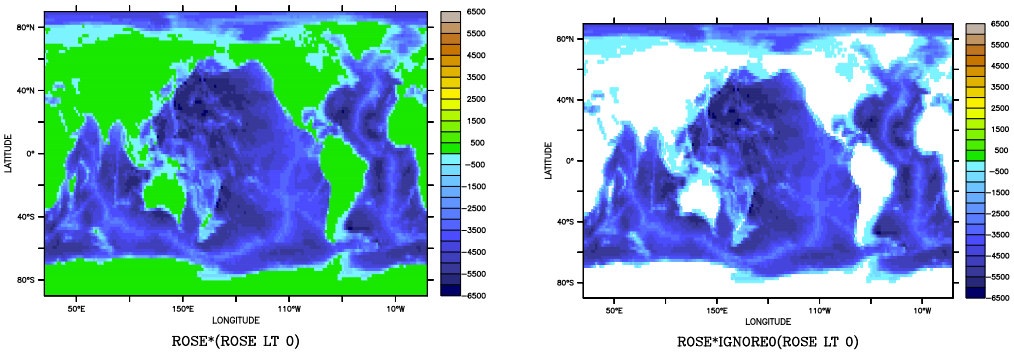
(b) Land-Meer-Maske (1=Ozean, 0=Land).

Abbildung 14: ETOPO und Ozean-Maske.

Diese beiden Felder (etopo und die 1/0 Maske) können nun mit einander verbunden (multipliziert) werden. Im einfachen Fall (15a), ist das Land jetzt einheitlich grün während der Ozean die gleichen Farbabstufungen hat wie in Abbildung 14a. Soll das Land jedoch ganz ausgeblendet werden, kann man die Funktion `IGNORE0(X)` verwenden (Abbildung Abbildung 15b).

```
yes? ! grünes Land:
yes? shade/pal=land_sea/lev=(-6500,6500,500) rose*(rose lt 0)

yes? ! kein Land:
yes? shade/pal=land_sea/lev=(-6500,6500,500) rose*IGNORE0(rose lt 0)
```



(a) Vollständige Topographie.

(b) Land-Meer-Maske (1=Ozean, 0=Land).

Abbildung 15: ETOPO und Ozean-Maske.

Variablen-Masking

Masken können auch verwendet werden, um Variablen zu definieren. Entweder wie beim Inline-Masking oder mit einem IF-THEN-ELSE Konstrukt:

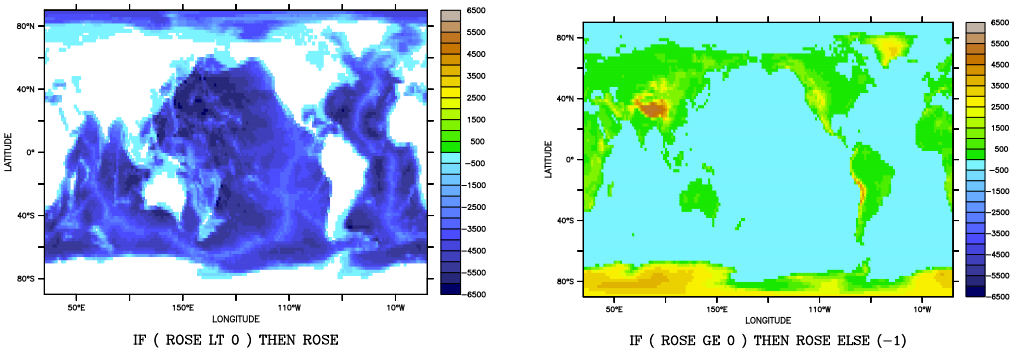
IF (<BEDINGUNG>) **THEN** <AUSDRUCK1>
oder
IF (<BEDINGUNG>) **THEN** <AUSDRUCK1> **ELSE** <AUSDRUCK2>

Beispiel 1: Als Beispiel dient wiederum der ETOPO-Datensatz. Als Alternative zum Kommando für Abbildung 15b wird nun eine neue Variable 'ocean' definiert, die überall den entsprechenden Wert der Variablen 'rose' haben soll, bis auf die Areale, in denen rose ≥ 0 ist. Ohne 'ELSE' wird der Landbereich mit missing values gefüllt (siehe Abbildung 16a):

```
yes? use etopo120
yes? let ocean = IF ( rose lt 0 ) THEN rose
yes? shade/pal=land_sea/lev=(-6500,6500,500) ocean
```

Um die Verwendung der 'ELSE' Angabe zu verdeutlichen wird nun das Land maskiert und der Ozean mit -1 gefüllt. Abbildung 16b zeigt daher das Land in den selben Farbabstufungen wie die ursprünglichen Abbildung 14a, der Ozean ist jedoch einheitlich in der ersten Blauabstufung dargestellt:

```
yes? let land = IF ( rose ge 0 ) THEN rose ELSE (-1)
yes? shade/pal=land_sea/lev=(-6500,6500,500) land
```



(a) Vollständige Topographie.

(b) Land-Meer-Maske (1=Ozean, 0=Land).

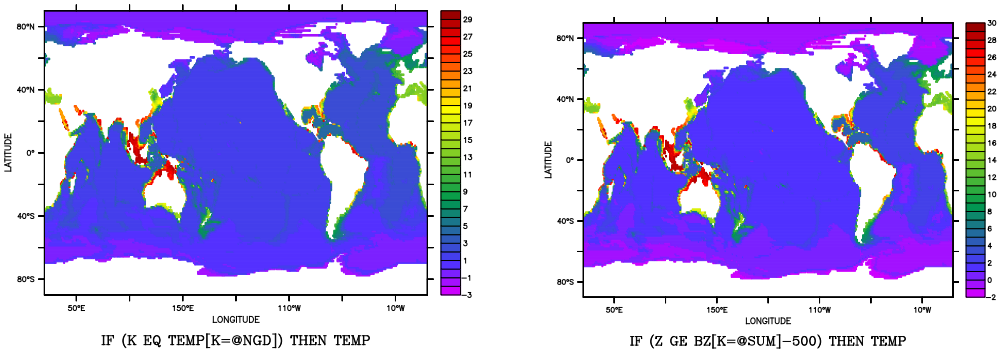
Abbildung 16: ETOPO und Ozean-Maske.

Beispiel 2: Aus dem Levitus-Datensatz wird zunächst die Bodentemperatur gesucht. Hierzu wird eine Variable **bt** definiert, die die Temperatur herauspickt, deren k-Index der Anzahl gültiger Werte auf der Z-Achse entspricht, also der tiefste gültige Wert. Da `bt` eine 3-dimensionale Variable sein wird mit jeweils nur einem gültigen Wert in der Tiefe, kann zum Plotten (Abbildung 17a) einfach die vertikale Summe (auch Mittelwert wäre möglich) gebildet werden:

```
yes? use levitus_climatology
yes? let bt = if (K eq temp[k=@NGD]) then temp
yes? ! (a)
yes? shade bt[k=@SUM]
```

Um die mittlere Temperatur einer 500m-Schicht über dem Boden zu ermitteln, ist eine Zwischenvariable (**bz**) nötig, nämlich die tatsächliche Tiefe der Bodenbox (basierend auf dem Gitter der Variable `temp`). Anhand dieser kann dann die 500m-Bodenschicht identifiziert werden und die Temperatur maskiert werden (**bt500**). Geplottet wird dann das vertikale Mittel (17b):

```
yes? let bz = if (K eq temp[k=@NGD]) then _z[g=temp]
yes? let bt500 = if (Z ge bz[k=@SUM]-500) then temp
yes? ! (b)
yes? shade bt500[z=@AVE]
```



(a) Temperatur der tiefsten Gitterbox.

(b) Mittlere Temperatur innerhalb 500m über dem Meeresboden.

Abbildung 17: Boden-Temperatur aus der Levitus Klimatologie.

5 Regridding

Eine weitere Stärke von Ferret ist die enge Verknüfpung von Daten und Gitter. So können mit wenig Aufwand eigene Achsen und Gitter definiert und Daten auf diese transformiert werden, das 'Regridding'.

5.1 Achsen definieren

Der Befehl zum Definieren neuer Achsen lautet `define axis` und steht in drei Varianten zur Verfügung:

- **define axis/<𝔊>=<START>:<END>:<DELTA>/<SPEC> <Achsen-Name>**
- **define axis/<𝔊>/<SPEC> <Achsen-Name> = <Ausdruck>**
- **define axis/<𝔊>/BOUNDS/<SPEC> <Achsen-Name> = <PUNKT-DATA>,<BOUND-DATA>**

<𝔊> steht dabei für eine der Dimensions-Achsen X,Y,Z,T,E oder F. Mithilfe der weiteren Kommando-Qualifier <SPEC> (siehe Tabelle 15) lassen sich genauere Angaben zur Achse machen, wie Werte-Bereich, Anzahl der Stützpunkte, Einheit, usw.

Die erste Variante des Kommandos nutzt die Angabe des Wertebereichs im Qualifier für die Dimensions-Achse; als Argument ist nur noch der Achsen-Name nötig. Die zweite Variante legt die Werte der Achse über einen Ausdruck fest, ähnlich einer Vairablenzuweisung, allerdings wird der Ausdruck sofort ausgewertet. Die letzte Form des Kommandos ist für Achsen reserviert, die irregulär sind und über sog. NetCDF Bounds festgelegt werden (wird hier nicht behandelt.)

Kommando-Qualifier	Erläuterung
/X/Y/Z/T/E/F	Gibt den Achsentyp an.
/DEPTH	Bei Z-Achsen kann hier die Richtung für positive Werte nach unten umgekehrt werden.
/MODULO	Wiederholende (Modulo) Achse, wie z.B. beim Längengrad
/NPOINTS	Anzahl der Datenpunkte, wenn kein <DELTA> angegeben wird
/T0	Referenz-Zeitpunkt, auf den sich das Datum einer Zeitachse bezieht. Default: 15-Jan-1901
/UNITS	Physikalische Einheit, die mit dieser Achse verknüpft ist: mm, cm, m, km, ft, in, mile, dbar, mb, level, layer, sec, min, hour, day, mon, yr (365 Tage), greogiran_year (365.2425 Tage), year360 (360 Tage), year366 (366 Tage), M2 cycles
/EDGES	Die Angaben für den Achsen-Bereich beziehen sich auf die Ränder, zwischen denen die Achsen-Werte angesiedelt werden, statt auf die Achsenwerte selbst.
/CALENDAR	Kalender, der der Zeitachse zu Grunde liegt: GREOGRIAN / STANDARD (365.2425 Tage), JULIAN (365.25 Tage mit Schalttagen), NOLEAP (365 Tage ohne Schalttage), 360_DAY (360 Tage, alle Monate 30 Tage)
/BOUNDS	Zum definieren von Achsen mithilfe von NetCDF Bounds.

Tabelle 15: Kommando-Qualifier für `define axis`

Raum

Beispiel 1: Längengrad Eine Längengrad-Achse mit 1° Gitterpunkt-Abstand

```
yes? define axis/x=0:360:1/units=longitude            lon1
yes? define axis/x/units=longitude                    lon2 = _X[x=0:360:1]
yes? define axis/x=0:360:1/units=longitude/modulo     lon3

yes? let lonvar = _X[X=1:360]*(3+COS((_I[I=1:360]-1)/180*3.14))/4
yes? define axis/x/units=longitude/mod                lon4 = lonvar

yes? sh ax lon1 lon2 lon3 lon4
 name       axis              # pts   start                end
 LON1      LONGITUDE          361 r   0E                   0E(360)
   Axis span (to cell edges) = 361
 LON2      LONGITUDE          361 r   0E                   0E(360)
   Axis span (to cell edges) = 361
 LON3      LONGITUDE          361mr   0E                   0E(360)
   Axis span (to cell edges) = 361 (modulo length = axis span)
 LON4      LONGITUDE          360mi   1E                   0.019W(359.98)
   Axis span (to cell edges) = 360.0038 (modulo length = axis span)
```

Beispiel 2: Druck-Tiefenachse Statt einer Tiefenachse in Meter soll eine Achse für den Druck (dbar) definiert werden:

```
yes? use levitus_climatology

yes? def ax/z/depth/units=dbar P=_Z[z=0:80:1]

yes?  sh ax p
 name       axis              # pts   start                end
 P         DEPTH (DBAR)        81 r-  0                    80
   Axis span (to cell edges) = 81

yes? def ax/z/depth/units=dbar TEMPonP=_Z[d=1,g=temp]*1.e-2

yes? sh ax TEMPonP
 name       axis              # pts   start                end
 TEMPONP   DEPTH (DBAR)        20 i-  0                    50
   Axis span (to cell edges) = 55.05
```

Zeit

Beispiel 1: Jährliche Achse Die neue Zeitachse soll zwischen 1958 und 2005 jeweils einen Eintrag pro Jahr haben und jedes Jahr soll 365 Tage lang sein (keine Schaltjahre):

```
yes? define axis/T=01-JAN-1958:31-DEC-2005/npoints=48/edg\
               /T0=01-JAN-1958/units=yr/cal=noleap annual

yes? show ax/l=1:3 annual
 name       axis              # pts   start                end
 ANNUAL    TIME                48 r   02-JUL-1958 11:45    01-JUL-2005 12:15
T0 = 01-JAN-1958
CALENDAR = NOLEAP
   Axis span (to cell edges) = 47.99726

       L     T                    TBOX      TBOXLO                 TSTEP (YR)
       1>  02-JUL-1958 11:45:00  0.9999429  01-JAN-1958 00:00:00      0.4999715
       2>  02-JUL-1959 11:15:00  0.9999429  31-DEC-1958 23:30:00      1.499914
       3>  02-JUL-1960 10:45:00  0.9999429  31-DEC-1959 23:00:00      2.499857
```

Beispiel 2: Saisonale Modulo-Achse Um klimatologische Mittel (jeweils gleiche Monate werden über mehrere Jahre hinweg gemittelt) aus einer Zeitreihe zu berechnen, kann eine sich wiederholende Klimatologische Zeitachse definiert werden. Ferret kommt bereits mit einigen klimatologischen Zeitachsen für unterschiedliche Kalender. Diese sind in einer NetCDF-Datei abgelegt, die aber sonst keine Daten enthält: `climatological_axes.cdf` (siehe Anhang C auf Seite 85).

```
yes? use climatological_axes
 *** NOTE: regarding /usr/local/PyFerret_1.0.2/go/climatological_axes.cdf ...
 *** NOTE: Climatological axes SEASONAL_REG, MONTH_REG, MONTH_IRREG, MONTH_GREGORIAN,
     MONTH_NOLEAP, MONTH_360_DAY, MONTH_ALL_LEAP  defined
yes? sh da
     currently SET data sets:
    1> /usr/local/PyFerret_1.0.2/go/climatological_axes.cdf  (default)
 name     title                               I         J         K         L
```

Ensemble/Forecast

Die Technik, in Ferret mehrere ähnliche Datensätze auf einer 'Ensemble-Achse' zusammenzufassen, um z.B. statistische Berechnungen durchzuführen ist sehr neu und noch in der Erprobung. Im Wesentlichen müssen in allen Datensätzen die gleichen Variablen enthalten sein (evt. müssen Datensatz-spezifische Variablen definiert werden, um diese Bedingung zu erfüllen; siehe 2.3) und die Variablen müssen auf dem gleichen Gitter definiert sein. Dann können sie mit folgendem Kommando zu einem Ensemble zusammengefasst werden:

define data/aggregate/E <Name> = <DATA> [**,<DATA>** [**,<DATA>** [**,...**]]]

oder kürzer:

ensemble <Name> = <DATA> [**,<DATA>** [**,<DATA>** [**,...**]]]

<NAME> ist der (kurze) Name des neuen Datensatzes (vergleichbar mit dem Dateinamen einfacher Datensätze). <DATA> [,<DATA> [,<DATA> [,...]]] ist eine, mit Kommas getrennte, Liste von Datensätzen, wobei <DATA> entweder eine Nummer eines bereits geladenen Datensatzes sein kann, oder der Name einer Datei. Die Online-Hilfe von Ferret[13] bietet hierzu ausführliche Beispiele.

Kommando-Qualifier	Erläuterung
/E	Ensemble immer entlang der E-Achse
/TITLE	Langer Datensatz-Name
/HIDE	Zeigt die zu Grunde liegenden Datensätze nicht im `show data` listing an.

Tabelle 16: Kommando-Qualifier für `define data/aggregate` (bzw. `ensemble`)

[13] http://ferret.pmel.noaa.gov/Ferret/documentation/users-guide/commands-reference/DEFINE#_define_data_agg

5.2 Gitter-Transformationen

Eine besondere Form der Transformation in Ferret ist die Gittertransformation, dabei werden Achsen oder ganze Gitter auf andere Gitter transformiert. Sie werden, wie Achsen-Transformationen, innerhalb eines Variablen-Qulaifiers festgelegt:

$$\textbf{\textit{variable}}[\ g\mathcal{X} = \texttt{<@RGR>}\ ]$$

$g\mathcal{X}$ steht dabei für eine der Gitterkomponenten `gx`, `gy`, `gz`, `gt`, `ge`, `gf`. Ferret bietet eine Vielzahl an Gitter Transfomrationen (Platzhalter `<@RGR>`; siehe Tabelle 17).

Beispiel 1: Horizontale Achsen

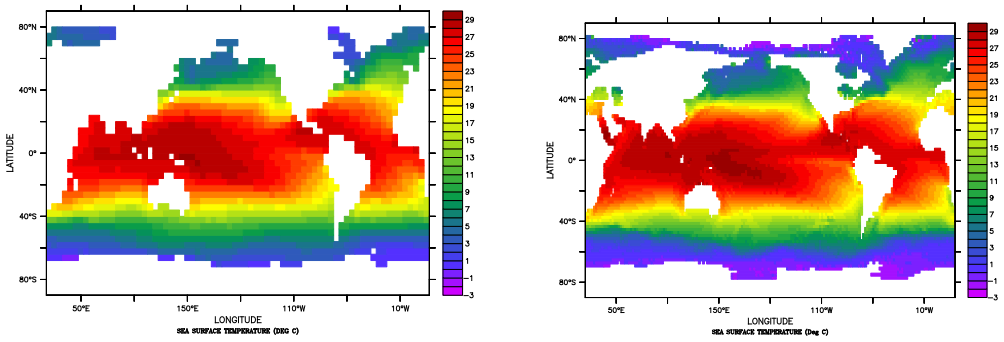
Als Beispiel soll die Differenz der Oberflächentemperatur (`sst`) zweier unterschiedlicher Datenprodukte ermittelt werden: aus ESKU und COADS. Ihre Gitter unterscheiden sich hauptsächlich in der horizontalen Auflösung und der Datenabdeckung.

```
yes? use esku_heat_budget
yes? use coads_climatology

yes? sh gr sst[d=1]
    GRID GQI1
 name       axis              # pts   start               end
 ESKUX     LONGITUDE          72mr   20E                 15E(375)
 ESKUY     LATITUDE           46 i   90S                 90N
 normal    Z
 TIME      TIME               12mr   16-JAN 06:00        16-DEC 01:20
 normal    E
 normal    F

yes? sh gr sst[d=2]
    GRID GSQ1
 name       axis              # pts   start               end
 COADSX    LONGITUDE          180mr  21E                 19E(379)
 COADSY    LATITUDE           90 r   89S                 89N
 normal    Z
 TIME      TIME               12mr   16-JAN 06:00        16-DEC 01:20
 normal    E
 normal    F

yes? ! (a)
yes? shade/lev=(-3,30,1) sst[d=1,t=@AVE]
yes? ! (b)
yes? shade/lev=(-3,30,1) sst[d=2,t=@AVE]
```



(a) ESKU (b) COADS

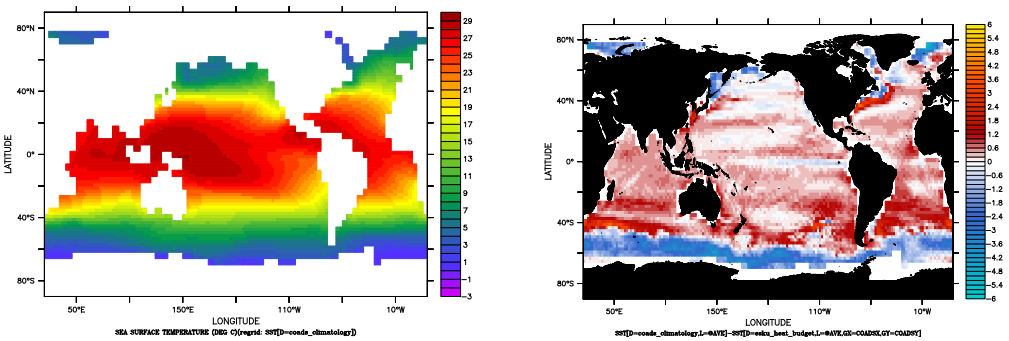
Abbildung 18: Meeresoberflächentemperatur (SST).

Aus den Abbildung 18a und 18b fällt auf, dass der COADS-Datensatz feiner aufeglöst ist und eine etwas größere Datenabdeckung hat. In Folge dessen wird die EKSU-SST zunächst auf das COADS-Gitter interpoliert (Abbildung 19a):

```
yes? shade/lev=(-3,30,1) sst[d=1,l=@AVE,gx=COADSX,gy=COADSY]
```

Nun kann die Differenz gebildet werden (Abbildung 19b):

```
yes? shade/lev=(-6,6,.2)/pal=centered_diff \
       sst[d=2,l=@AVE]-sst[d=1,l=@AVE,gx=COADSX,gy=COADSY]
yes? go fland 5 black
```



(a) ESKU Werte auf COADS-Gitter interpoliert.

(b) Differenz: COADS-ESKU

Abbildung 19: Meeresoberflächentemperatur (SST).

Transformation <@RGR>	**Erläuterung**
@LIN	Lineare Interpolation zwischen Ursprungs- und Ziel-Gitter
@AVE	Gewichtetes Mittel. Bei mehreren Achsen gleichzeitig angewendet, werden die Transformationen aufeinander aufbauend ausgeführt.
@ASN	Ungewichtete Ersetzung der Achsenwerte
@VAR	Varianz der Datenpunkte auf der Ursprungsachse, die in eine Ziel-Gitterbox fallen.
@NGD	Anzahl der gültigen Werte, die in eine Ziel-Gitterbox fallen.
@NRST	Wählt den Wert, dessen ursprünglicher Achse-Wert dem Zielachsen-Wert am nächsten liegt.
@SUM	Bildet die Summe über alle Werte innerhalb einer Ziel-Gitterbox
@MIN	Ermittelt das Minimum über alle Werte innerhalb einer Ziel-Gitterbox
@MAX	Ermittelt das Maximum über alle Werte innerhalb einer Ziel-Gitterbox
@XACT	Übernimmt nur diejenigen Werte, deren ursprünglichen Achsen-Werte eine exakte Entsprechung auf der Zielachse finden, alle anderen werden auf 'missing value' gestzt.
@MOD	Berechnet eine Klimatologie entlang einer Modulo-Ziel-Zeitachse. Der Kalender zwischen alter und neuer Achse muss übereinstimmen.

Tabelle 17: Gitter-Transformationen.

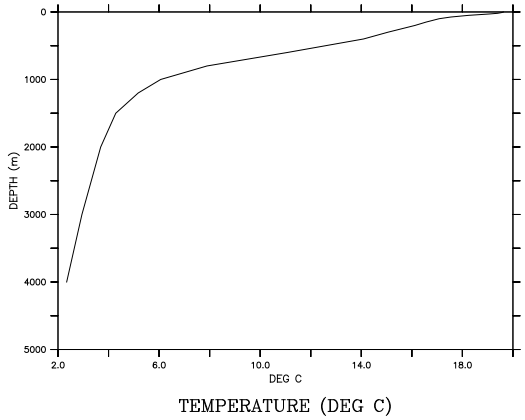
Beispiel 2: Vertikale Achse

Das vertikale Temperatur-Profil bei 40°N und 40°W aus Levitus soll, statt auf einer Tiefenachse in Meter (siehe Abbildung 20a), auf einer Druckachse in Dezibar (dbar) geplottet werden. Hierzu wird zunächst, wie im Abschnitt oben, eine einfache Druckachse aus den Tiefeninformationen generiert (1m $\sim$ 0.01dbar). Anschließend wird das Temp-Profil mit Standard-Interpolation (Linear) bzw. ohne Interpolation (@ASN) auf diese Druckachse transformiert (vgl. Abbildung 20b):

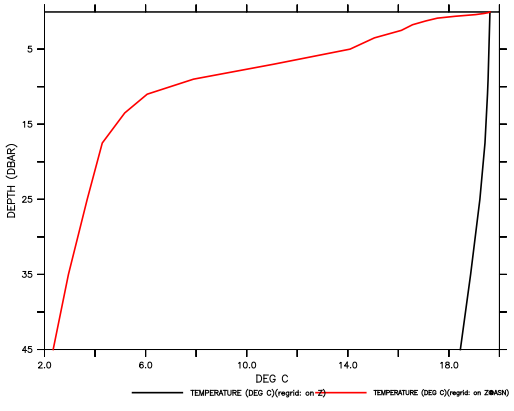
```
yes? use levitus_climatology
yes? ! (a)
yes? plot temp[x=40W,y=40N]

yes? ! (b)
yes? def ax/z/depth/units=dbar TEMPonP=_Z[d=1,g=temp]*1.e-2
yes? plot/thick=2 temp[x=40W,y=40N,gz=temponp], \
                  temp[x=40W,y=40N,gz=temponp@ASN]
```

Es fällt auf, dass der erste Ausdruck im Plotkommando (`temp[x=40W,y=40N,gz=temponp]`; schwarze Linie in Abbildung 20b) die falschen Temperatur-Werte ausgibt. Das liegt an der Interpolation, die **standardmäßig**, also ohne Angabe irgend eines <@RGR>-Operators, **linear** ist. Dabei werden nur die oberen 50 (Meter) aus dem ursprünglichen Datensatz berücksichtigt, weil die Druckachse nur bis 50 (dbar) geht und Ferret beim linearen Interpolieren nur die reinen Zahlenwerte vergleicht. Wird hingegen die @ASN Transformation eingesetzt, werden die Temperatur-Werte nur entsprechend ihrem Index auf die neue Achse übertragen, ungeachtet der Achsen-Werte und das Profil wird richtig ausgegeben (rote Linie).



(a) Temperatur der tiefsten Gitterbox.



(b) Mittlere Temperatur innerhalb 500m über dem Meeresboden.

Abbildung 20: Boden-Temperatur aus der Levitus Klimatologie.

Zeit-Achse

Eine Zeitserie (`moc_timeseries.nc`) mit monatlichen Werten zwischen 1958 und 2005 soll mit unterschiedlichen Transformationstechniken auf eine neue Zeitachse mit jährlichen Einträgen gerbracht werden, d.h. es sollen die Jahres-Mittel geplottet werden (siehe Abbildung 21):

```
yes? use moc_timeseries

yes? define axis/T=01-JAN-1958:31-DEC-2005/npoints=48/edg\
                /T0=01-JAN-1958/units=yr/cal=noleap annual

yes? plot/color=(80,80,80)/hlim=01-JAN-1958:31-DEC-1964/vgrat moc
```

Die ursprüngliche (monatliche) Zeitreihe wird als graue Linie dargestellt.

(1) 12-Monatsmittel auf orginal-Zeitachse:

```
yes? plot/ov/col=black/thick=1/sym=18/line moc[l=1:576:12@AVE]
```

Für den ersten Wert werden die ersten 6 Werte von 1958 gemittelt. Anschließend werden die nachfolgenden 12 Werte gemittelt, also L=1:6, L=7:18, L=19:30 usf. Die resultierenden Werte repräsentieren die Mittel Juli-Juni.

(2) Boxfilter über 25 Einträge auf monatlicher Achse:

```
yes? plot/ov/col=blue/thick=1/sym=19/line moc[l=@SBX:25]
```

Bei dieser Transformation wird ein Fenster mit einer Breite von 25 (monatlichen) Zeitpunkten über die Zeitreihe geschoben, jeweils um einen Monat versetzt und das Mittel ausgegeben. Daher werden erst Werte ab 01-JAN-1959 ausgegeben. Dies entspricht einem ungewichteten Recheck-Filter.

(3) Hanningfilter über 25 monatliche Einträge:

```
yes? plot/ov/col=(0,80,0)/thick=1/sym=20/line moc[l=@SHN:25]
```

Ähnlich dem vorhergehenden Recheckfilter, wird auch hier ein Fenster über die Zeitreihe geschoben und gemittel. Es wird jedoch mit der Mitte des Fensters gewichtet, sodass die Zeitreihe nahezu dem einfachen 12-Monats-Mittel entspricht.

(4) Lineare Interpolation auf die Jahres-Zeitachse

```
yes? plot/ov/col=(100,55,0)/thick=1/sym=21/line moc[gt=annual]
```

Das Ergenis dieser Transformation fällt aus der Reihe. Statt Mittelwerte, wurden nur die Zeiteinträge der ursprünglichen Achse berücksichtigt, die am nächsten zum neuen Zeitpunkt auf der jährlichen Achse liegen.

(5) Einträge werden für neue Zeitachse gemittelt

```
yes? plot/ov/col=(100,0,0)/thick=1/sym=22/line moc[gt=annual@AVE]
```

Um die Werte der alten Zeitreihe auf die Jahres-Einträge der neuen Achse tatsächlich zu mitteln, ist die Gitter-Transformation @AVE nötig. Es fällt auf, dass die resultierenden Werte (rote Sternchen und die rote Linie in Abbildung 21) sich zum Teil erheblich von der gefilterten Zeitreihe unterscheiden.

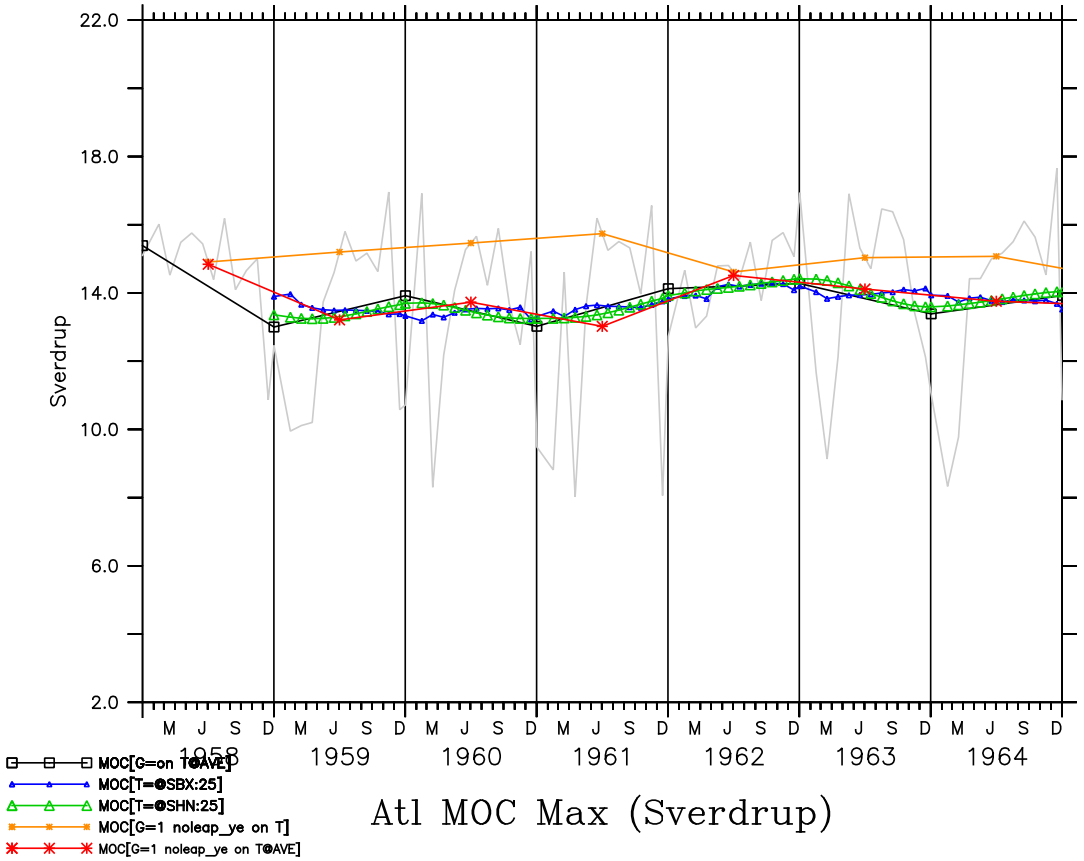


Abbildung 21: MOC Zeitreihe: Monatliche Werte (grau) und verschiedene Transformationen: [L=1:576:12@AVE] (schwarz), Rechteckfilter (blau), Hanning-Filter (grün), 'regriddded' auf jährliche Zeitachse ohne Interpolation (orange) und mit Mittelungungsoperator (rot).

6 Grafische Ausgabe anpassen

6.1 Das Grafik-Fenster

In einer interaktiven PyFerret-Sitzung wird ein Grafikfenster geöffnet, sobald ein Plotkommando ausgeführt wird. Sie werden entsprechend ihrer Reihenfolge, wie sie erzeugt wurde, durchnummeriert. Abbildung 22 zeigt die einzelnen Elemente eines solchen Fensters:

- Plotbereich
- Achsen, deren Werte und Beschriftung
- Titel
- Farbbalken
- zusätzliche Label

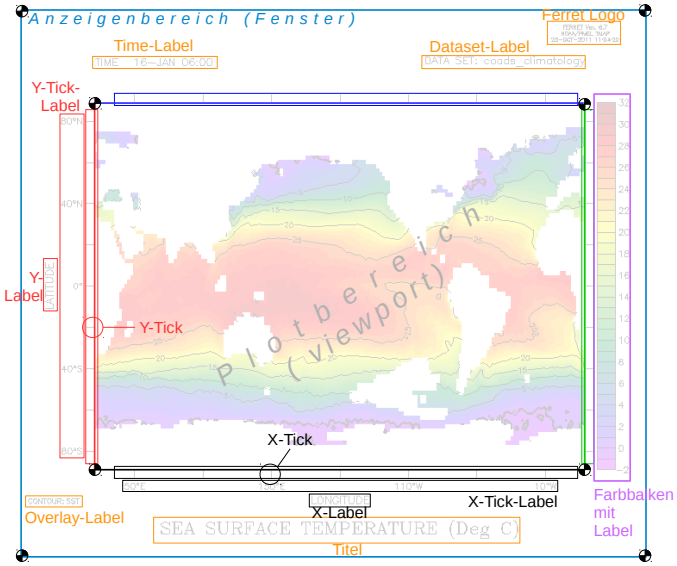


Abbildung 22: Das Ferret Grafik-Fenster

Innerhalb eines Grafik-Fensters gibt es zwei Modi: (1) Die Grafik wird im gesamten Fenster dargestellt oder (2) der User wählt zuvor einen bestimmten Viewport (eine Art Zeichenbereich innerhalb des Fensters; siehe 6.2), mit deren Hilfe auch mehrere Grafiken in einem Fenster untergebracht oder übereinandergelegt werden können. In beiden Fällen verteilt Ferret selbständig die Elemente.

Es stehen drei Kommandos für die Arbeit mit Fenstern zur Verfügung:

```
show window [/all]
set window/<qualifier>  [<Fenster-NR>]
cancel window [/all]  [<Fenster-NR>]
```

Das erste Kommando, `show window`, zeigt alle bislang geöffneten Fenster samt ihrere Eigenschaften an. Diese Eigenschaften können mit dem zweiten Kommando, `set window`, verändert werden. Es dient auch dazu Fenster mit bestimmten Eigenschaften neu anzulegen. Überflüssige Fenster können mit `cancel window` kontrolliert geschlossen werden (beachte Info 7)

Fenstereigenschaften

Zu den Eigenschaften eines Fensters gehören das Seitenverhältnis, die Ausgabequalität, die Größe der dargestellten Grafik (*nur in PyFerret*) und des Fensters sowie die Hintergrundfarbe. Sie können mit dem Kommando

```
set window/<qualifier> | <Fenster-NR>
```

verändert werden. Tabelle 18 listet die entsprechenden /<qualifier> auf. Wird die <Fenster-Nr> weggelassen, beeinflussen die Änderungen nur das aktuelle Fenster; ist noch keines vorhanden muss es mit dem Qualifier /new erzeugt werden. Im folgenden werden einige dieser Eigenschaften näher erläutert.

Kommando-Qualifier	Erläuterung
/new	Neues Fenster erzeugen
/size=<r>	Fenstergröße
/aspect=<r>	Seitenverhältnis
/xinches=<r>	Breite der Grafik in Inch (nicht des Fensters)
/yinches=<r>	Höhe der Grafik in Inch (nicht des Fensters)
/xpixels=<n>	Breite der Grafik in px (nicht des Fensters)
/ypixels=<n>	Höhe der Grafik in px (nicht des Fensters)
/title="<text>"	Fenstertitel
/location=<x>,<y>	Position der unteren linken Ecke auf Bildschirm (Werte zwischen 0 und 1)
/clear	Fensterinhalt löschen (alternative: cancel viewport)
/quality={high,draft}	Qualität: draft=Arbeitsqualität, high=publizierbar
/[non]antialias	Antialiasing
/color=<color>	Fensterhintergrund (<color> kann Farb-Nr., -Name oder ein (R,G,B) Tripel sein.)

Tabelle 18: Kommando-Qualifier für Kommando set window.

Info 7: Schließen des Grafik-Fensters:

PyFerret Im Allgemeinen kann das Grafikfenster in Pyferret mit der Maus über das Fenstermenü, dem × im Fensterrahmen oder durch ein Kommando geschlossen werden. Wird eine grafische Ausgabe erzeugt, öffnet sich ein neues Fenster.

Ferret Im Gegensatz zu PyFerret sollte in Ferret das Fenster **niemals** über das × im Fensterrahmen des Window-Managers geschlossen werden, denn das kann zum Absturz der gesamten Ferret-Session führen. Stattdessen kann mit einem Kommando das Fenster ordentlich geschlossen werden und bei Bedarf mit einem andere Kommando ein neues wieder geöffnet werden (Das erste Fenster einer Session öffnet sich beim ersten Plot-Kommando automatisch).

Größe und Seitenverhältnis Die Größe bzw. Skalierung des Fensters (nicht der Grafik) kann mit `/size=<Scale>` gesetzt werden. Unabhängig von der Fenster-Skalierung ist die Größe der Grafik, wenn sie als Datei abgespeichert wird. Diese kann mit `/xinches=<X>` und `/yinches=<Y>` oder `/xpixels=<X>` und `/ypixels=<Y>` gesetzt werden.

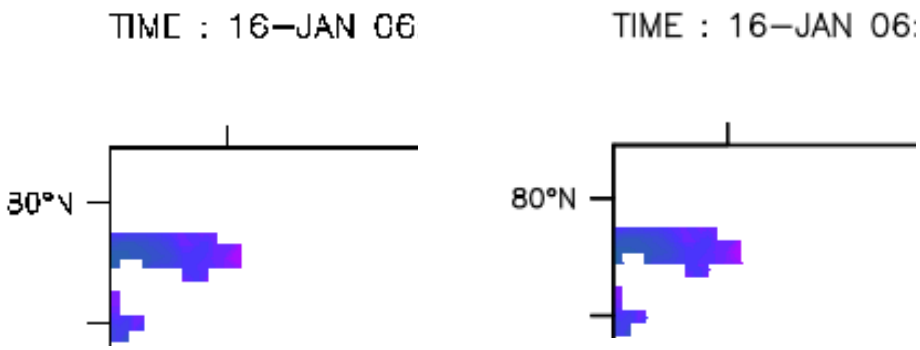
Steht das Seitenverhältnis (siehe Abbildung 23) im Vordergrund, kann der Qualifier `/aspect=<ratio>` gesetzt werden. Allerdings können auch Kombinationen aus Seitenverhältnis und einer Seitenlänge sinnvoll sein. Einige Beispiel:

```
yes? set window/aspect=0.6/size=1 1
yes? set window/aspect=0.6/xinches=11/size=2 2
yes? set window/xinches=11/yinches=6.6 3
yes? set window/xpixels=800/ypixels=600 4
```

Hinweis für PDF-Ausgabe: Wird eine Grafik über das Kommando frame als PDF abgespeichert, entspricht die 'Media Size' immer dem Format Letter (8.5×11 inch). Es kann vorkommen, dass bei sehr schmalen Darstellungen, die Grafik auf dem Blatt abgeschnitten wird. Abhilfe schafft dann die explizite Angabe der längsten Seitenlänge der Grafik von **11 inch** mittels Qualifier `/xinches=<X>` oder `/yinches=<Y>`.

Abbildung 23: Seitenverhältnis (`set window/aspec`).

Bitmap-Qualität Mit PyFerret wurden einige Verbesserungen für die Qualität der grafischen Ausgabe in Dateien eingeführt. Hiervon profitieren vor allem Bitmap-Grafiken (nicht Postscript), wie der Vegleich von Abbildung 24a und 24b zeigt. Der entsprechende Qualifier hierfür lautet `/quality=`. Mit der zus. Angabe '`draft`' ist die Datei zwar klein und der Aufbau der Grafik schneller, es können jedoch Artefakte bzw. Fehldarstellungen besonders bei schmalen Strichen und Text auftauchen. Mit dem Wert '`high`' geht ein Antialiasing einher, mit dessen Hilfe auch Pixel-Grafiken gut skaliert werden können, ohne die Pixeldichte zu erhöhen. Die Darstelllung zusätzlicher Farbwerte vergrößert jedoch die Dateigröße.



(a) Draft (b) High

Abbildung 24: Qualität der Grafikausgabe (Qualifier `set window/quality`).

6.2 Viewports

Zu Beginn des Abschnittes 6.1 ist schon das Konzept der Viewports angeklungen. Im Wesentlichen kann man diese als Unterteilung des Fensters in verschiedene Darstellungsbreiche verstehen, die sich auch überlagern können. Es gibt eine Reihe vordefinierter Viewports, die mit dem Kommando

```
show viewport/all
```

aufgelistet werden können (vgl. hierzu Abbildung 25). Ihre Größe und Lage wird über die Position der unteren linken und oberen rechten Ecke relativ zum Fensterrahmen definiert. Dabei gibt es zwei Modi, in denen die Viewports definiert werden können: *edges* und *axes*. Im ersten Fall beziehen sich die Ecken auf die Außenkanten der Grafik inklusive Beschriftungen und Farbbalken. Bei letzerem wird die Positionen der Achsen-Ecken ohne Beschriftung usf. festgelegt.

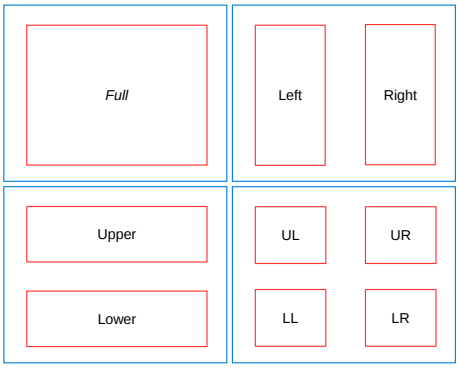


Abbildung 25: Verschiedene Viewports. Die blauen Rahmen markieren jeweils den Anzeigenbereich eines Fensters. Dir roten Rahmen sind vordefinierte Viewports. *Full* is der Standard-Viewport.

Viewport verwenden

Um einen bestimmten Viewport zu verwenden, muss vor dem ersten Zeichenbefehl das Kommando

```
set viewport <ViewPortName>
```

mit der entsprechenden Viewport-Bezeichnung für `<ViewPortName>` ausgeführt werden. Wurde der Viewport zwischenzeitlich gewechselt, muss die Grafik darin nochmals neu begonnen werden (d.h. der erste Plotbefehl ohne /overlay Qualifier, da es sonst zu Fehldarstellungen kommen kann, weil Ferret sich immer nur die Characteristiken der letzten Plotkommandos merkt, ungeachtet des Viewports.)

Neue viewports festlegen

Mit dem Kommando define viewport können neue Viewports definiert werden:

```
define viewport/<qualifier>
```

Die möglichen Kommando-Qualifier sind in Tabelle 19 aufgeführt.

Kommando-Qualifier	Erläuterung
/xlimits=<r>,<r>	X-Werte der unteren linken und oberen rechten Ecke ($0 \leq r \leq 1$).
/ylimits=<r>,<r>	Y-Werte der unteren linken und oberen rechten Ecke ($0 \leq r \leq 1$).
/axes	axes Modus, sonst edges Modus.
/text=<r>	Relative Textgröße (Default: 1)

Tabelle 19: Kommando-Qualifier für Kommando `define viewport`.

Das Kommando

cancel viewport <ViewPortName>

hingegen dient dazu, den entsprechenden Viewport <ViewPortName> wieder zu löschen.

Größe und Lage Die Qualifier /xlimits und /ylimits erwarten (mit Komma getrennt) je zwei Werte: Die X- bzw Y-Position der unteren linken und der oberen rechten Ecke. Die Werte liegen zwischen 0 und 1 und beziehen sich auf das Fenster. Im Standard-Modus 'edge' muss der Benutzer nicht selbst den Platz für Titel, Achsenbeschriftung und Farbbalken berücksichtigen.

Eine elegante Möglichkeit, viewports zu definieren, bietet das Skript portraitNxN <H> <V>, mit dessen Hilfe sich eine Matrix aus gleichverteilten H × V Viewports generieren lässt (H (V) ist die Anzahl der Viewports in der Horizntalen (Vertikalen)). Die einzelnen Viewports werden mit 1 beginnend (oben links) durchnummeriert und können über diese Nummer auch angesprochen werden. Um beispielsweise 6 Viewports (2 × 3) anzuordnen (siehe Abbildung 26) genügt die Eingabe:

```
yes? go portraitNxN 2 3
```

Anschließend können die einzelnen Unter-Grafiken in den Viewports erzeugt werden:

```
yes? use coads_climatology
yes? se vi 1
yes? shade sst[l=1]
yes? se vi 2
yes? !...
```

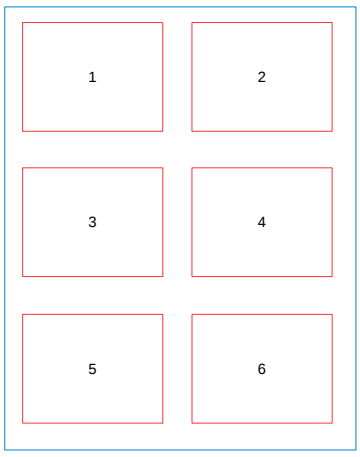


Abbildung 26: Viewports durch den Skriptaufruf "`go portraitNxN 2 3`" generiert.

Modus Edge vs. Axes Die Wirkungsweise des Qualifiers /axes lässt sich am einfachsten mit dem Standard-Viewport *Full* verdeutlichen. In Abbildung 27 markieren die roten Boxen die Lage der Achsen. Im edge-Modus wird genügend Platz um die rote Box gelassen, um Beschriftungen usw. unterzubringen. Im axes-Modus hingegen liegen die Achsen auf dem Rand des Fensters. Abbildung 28 zeigt die Positionsangaben für die Standard-Viewports, würde man sie im axes-Modus definieren. Dabei muss der Benutzer den Platz für Beschriftung, Titel und Farbbalken selbst berücksichtigen.

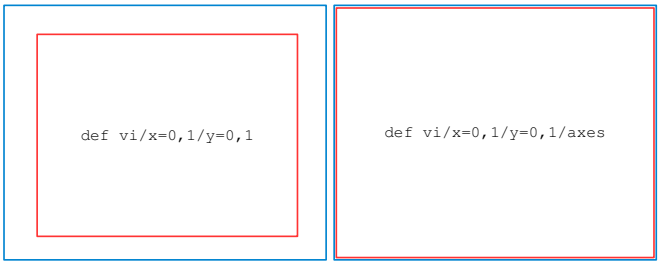


Abbildung 27: *Full* Viewport im edge-Modus (links) und im axes-Modus (rechts)

Der große Vorteil des axes-Modus zeigt sich aber erst, wenn viewports aneinandergehaftet werden, um bestimmte Bereiche einer Grafik z.B. mit abweichenden Achsen-Skalierungen darzustellen. Siehe hierzu Beispiel 5 und Abbildung 29, wo die oberen 500m der meridionalen Stromfunktion vergrößert dargestellt werden (Tatsächlich sind im Beispielcode zwei Darstellungsweisen umgesetzt. In der ersten Variante bleibt zwischen den Viewports ein weißer Strich zu sehen, da die Interpolation der Farbfelder nicht unter den unsichtbaren Axen-Rahmen reicht. Im zweiten Fall wird zunächst mit `shade` explizit ein größerer Tiefenbereich geplottet (z.B. `z=0:600`), als auf der Y-Achse dargestellt werden kann: `vlim=0:500` und dann ein `fill` darübergelegt).

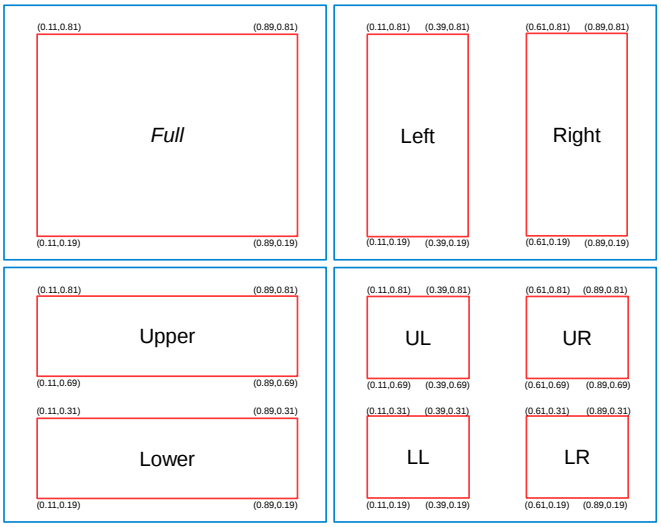


Abbildung 28: Standard Viewports mit Positionsangaben im axes-Modus.

Beispiel 5

```
use Atlantic_moc.nc

def vi/x=0.11:0.89/y=0.19:0.81/ax/text=1 UB
def vi/x=0.11:0.89/y=0.5:0.81/ax/text=1 uh
def vi/x=0.11:0.89/y=0.19:0.5/ax/text=1 bh

se wi/asp=.6/xinch=11/qual=high
se vi ub
fill/vlim=6000:500:-500/hlim=30S:90N/lev=(-14,14,2)/ax=(1,1,0,0)/z=400:6000 \
                zomsfatl+1000,nav_lat,z[g=zomsfatl]
se vi bh
fill/vlim=6000:500:-500/hlim=30S:90N/lev=(-14,14,2)/nolab/nokey\
                /ax=(0,0,1,1)/z=400:6000 zomsfatl,nav_lat,z[g=zomsfatl]
se vi uh
fill/vlim=500:00:-100/hlim=30S:90N/lev=(-14,14,2)/nolab/nokey\
                /ax=(0,0,1,1)/z=0:600 zomsfatl,nav_lat,z[g=zomsfatl]


!===== Mit unterlegtem  shade (kein weiß er Strich): =====

se vi ub
fill/vlim=6000:500:-500/hlim=30S:90N/lev=(-14,14,2)/ax=(1,1,0,0)/z=400:6000 \
                zomsfatl+1000,nav_lat,z[g=zomsfatl]
se vi bh
shade/trim/vlim=6000:500:-500/hlim=30S:90N/lev=(-14,14,2)/nolab/nokey\
                /ax=(0,0,1,1)/z=400:6000 zomsfatl,nav_lat,z[g=zomsfatl]
fill/ov/vlim=6000:500:-500/hlim=30S:90N/lev=(-14,14,2)/nolab/nokey\
                /ax=(0,0,1,1)/z=400:6000 zomsfatl,nav_lat,z[g=zomsfatl]
se vi uh
shade/trim/vlim=500:00:-100/hlim=30S:90N/lev=(-14,14,2)/nolab/nokey\
                /ax=(0,0,1,1)/z=0:600 zomsfatl,nav_lat,z[g=zomsfatl]
fill/ov/vlim=500:00:-100/hlim=30S:90N/lev=(-14,14,2)/nolab/nokey\
                /ax=(0,0,1,1)/z=0:600 zomsfatl,nav_lat,z[g=zomsfatl]
```

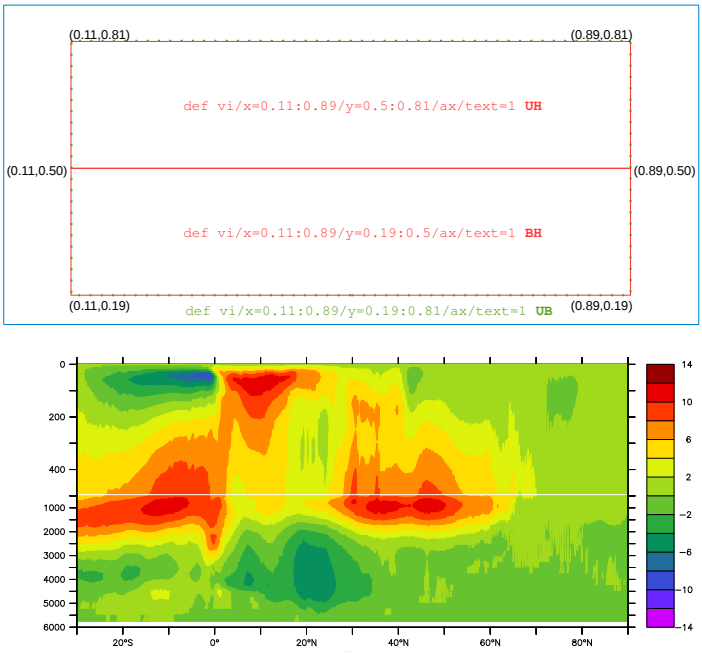


Abbildung 29: Oben: Aneinandergestellte Viewports (oberer und unterer Viewport als rote Boxen; full Viewport grün gepunktet; Positionen der Eckpunkte als (X,Y) Parre). Unten: Mittlere MOC im Atlantik. Die oberen 500m sind gedehnt im oberen Viewport dargestellt.

6.3 Grafikausgabe in Datei

Grafikdateien können in Ferret mit Hilfe des Kommandos frame erzeugt werden, nachdem alle gewünschten Plot-Kommandos ausgeführt wurden.

```
frame/<qualifier>
```

Die Angabe der passenden Dateinamens-Erweiterung in /file= (siehe Tabelle 20) ist ausreichend, so dass i.A. das Format nicht explizit angegeben werden muss.

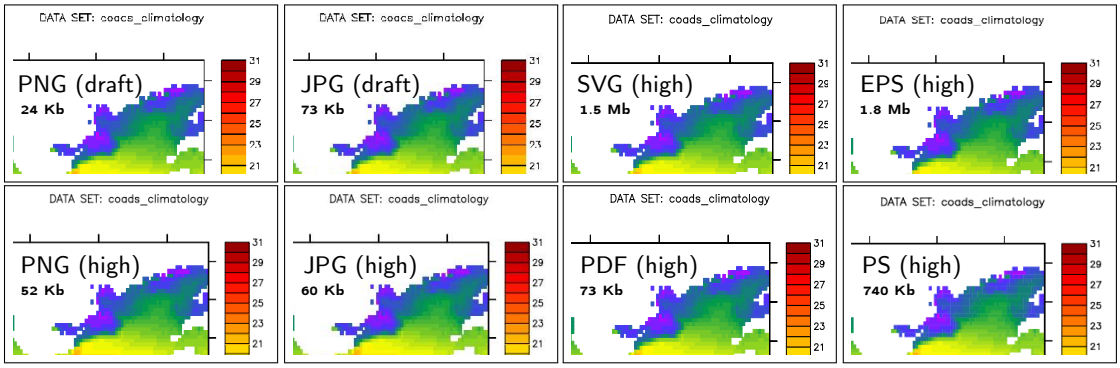
Kommando-Qualifier	Erläuterung
`/file="<DateiName>"`	Name der Grafikdatei
`/format=<DateiFormat>`	Datei-/Grafikformat
`/transparent`	Transparenter Hintergrund

Tabelle 20: Kommando-Qualifier für Kommando `define viewport`.

Tabelle 21 listet die möglichen Dateiformate unter Berücksichtigung der gewählten Grafik-Qualität. Die Formate png (gif) und jpg sind reine Bitmap-Formate und lassen sich gewöhnlich nur eingeschränkt skalieren; svg, ps, eps[14] und pdf (sowie das alte Meta-Format plt) sind Vektor-Formate und daher prinzipiell sehr gut zum Skalieren geeignet. Bei großer Farbzahl und vielen unterschiedlichen Flächen können diese Formate jedoch leicht mehrere Gb an Dateigröße einnehmen.

	Bitmap			**Vektor**				
/quality=draft	~~gif~~	*png*	*jpg*					
/quality=high		*png*	*jpg*	~~plt~~	*svg*	ps	*eps*	*pdf*

Tabelle 21: Grafikqualität und Dateiformate. *Kur*siv gedruckte Formate sind nur in PyFerret möglich. ~~Durchgestrichene~~ Formate waren in Ferret üblich und sind in PyFerret nicht mehr vorhanden, wobei Ferret nicht zwischen 'draft' und 'high' Qualität unterschieden hat.



(a) Bitmap (b) Vektor

Abbildung 30: Verschiedene Grafikdateiformate

[14] eps Grafiken sind immer aufrecht und unabhängig von Papiergrößen

6.4 PlotPlus (PPL)

Neben den in Kapitel 3 bereits vorgestellten Plotkommandos bietet Ferret eine Erweiterung mit Namen *PlotPlus,* mit dessen Hilfe das Aussehen einer Grafik zusätzlich verändert werden kann. Die zugehörigen Kommandos sind Subkommandos von `ppl`:

ppl <ppl-Kommando> [**,<spec>** [**,<spec>**[**,...**]]]

Alle weiteren Angaben (<spec>) werden mit einem vorangestellten Komma aufgelistet. Dabei muss jedoch die Reihenfolge eingehalten werden. Soll eine <spec>-Angabe übersprungen werden, wird nur ein Komma angegeben.

Die meisten PPL Kommandos beziehen sich direkt auf Elemente eines bereits erfolgten Plot-Kommandos. In diesen Fällen werden PPL Instruktionen daher von einem plot-Befehl so eingerahmt, dass das Plot-Kommando mit dem Qualifier `/setup` eingeleitet wird (es wird noch nichts in das Fenster/den Viewport gezeichnet), mit ppl-Befehlen modifiziert und einem passenden `ppl <PlotBefehl>` abgeschlossen wird:

	Plot mit PPL	*Fill mit PPL*
Plot-Einleitung	**plot**/set	**fill**/set
PPL Kommandos	ppl ...	ppl ...
	ppl ...	ppl ...
Plot-Abschluss	ppl **plot**	ppl **fill**

Anpassungen von Grafikelementen

Titel Statt /title=" " im plot-Kommando. Erlaubt eine Angabe der Textgröße sowie Zeilenumbrüche:

ppl title,<height>,<label>

<height>	Höhe des Titels in Inch.
<label>	Titel-Text evt. mit Farb-Angaben (@Cnnn) oder Pen-Angaben (@Pn) oder Zeilenumbrüchen (<NL>)

```
use coads_climatology
shade/setup sst[l=1]
  ppl title,0.16,@P7Oberflaechentemperatur<NL>@P1Januar
ppl shade
```

Achsen-Typ Eine Grafik-Achse kann normal oder logarithmisch dargestellt werden:

ppl axtype,<typeX>,<typeY>

<typeX>	Achsen-Typ:
<typeY>	1: normal
	2: logarithmisch
<label>	Titel-Text evt. mit Farb-Angaben (@Cnnn) oder Pen-Angaben (@Pn) oder Zeilenumbrüchen (<NL>)

```
use coads_climatology
plot/set sst[y=@AVE,t=@VAR]
  ppl axtype,1,2
ppl plot
```

Achsen-Beschriftung Hiermit können die direkten Achsenbeschriftungen neu gesetzt werden:

ppl xlab,<height>,<labelX>
ppl ylab,<height>,<labelY>

<typeX>	Achsen-Typ:
<typeY>	1: normal
	2: logarithmisch
<label>	Titel-Text evt. mit Farb-Angaben (@Cnnn) oder Pen-Angaben (@Pn) oder Zeilenumbrüchen (<NL>)

```
use coads_climatology
shade/set sst[l=1]
  ppl xlab,.13,Laengengrad
  ppl ylab,.13,Breitengrad
ppl shade
```

Farbbalken anpassen Um die Ausrichtung und Größe sowie die Beschriftungen des Farbbalken zu ändern:

ppl shakey ,1<DoKey> ,2<Orient> ,3<KLabSiz> ,4<KLabInc> ,5<KLabDig> ,6<KLabLen> ,7<KxLo> ,8<KxHi> ,9<KyLo> ,10<KyHi>

1<DoKey>	Farbbalken ein (1) oder aus (0)
2<Orient>	Horizontaler (0) vertikaler (1) Farbbalken
3<KLabSiz>	Schriftgröße der Label am Balken in Inch
	(0: automatisch)
4<KLabInc>	Inkrement der Label am Farbbalken
	(0: automatisch)
5<KLabDig>	Zahl der Dezimalstellen in Labels (>0) oder 10er Stellen (<0)
	sonst 3 Stellen
6<KLabLen>	Maximale Zahl an Zeichen im Label
7<KxLo>	X-Koordinate linke Seite
8<KxHi>	X-Koordinate rechte Seite
9<KyLo>	Y-Koordinate Unterkante
10<KyHi>	Y-Koordinate Oberkante

```
use coads_climatology
shade/set sst[t=@AVE]
  ppl shakey,1,0,,,,
ppl shade
```

Um die aktuellen Einstellungen des letzten Plot-Befehls einzusehen, steht das folgende Kommando zur Verfügung:

ppl list shakey

```
yes? ppl list shakey

         DO KEY  ORIENT  LAB SIZE  LAB INC  LAB DIG  LAB LEN
              1       0      0.00        0        0        0


         DEFAULT KEY POSITIONING
         X LO     X HI     Y LO     Y HI
         0.00     0.00     0.00     0.00
```

Zusätzliche Beschriftungen

Grafiken können durch zusätzliche Beschriftungen (sog. Label) ergänzt werden und müssen nicht von einem `<plot>/set ... ppl <plot>` eingerahmt werden:

```
ppl %label /nouser ,<xPos> ,<yPos> ,<Align> ,<Angle> ,<Size> ,<Text>
                                oder
label              ,<xPos> ,<yPos> ,<Align> ,<Angle> ,<Size> ,<Text>
```

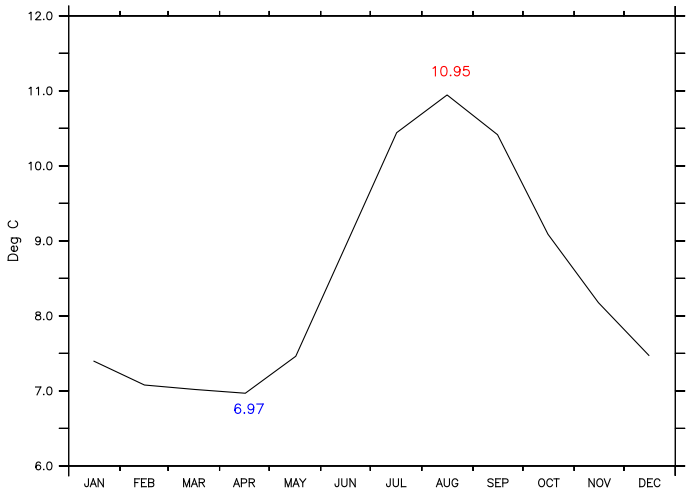
/nouser	Positionsangaben in Inch nicht in Achsen-Koordinaten.
<xPos>	Position des Label-Ankers in Achsenkoordinaten oder Inch
<yPos>	
<Align>	-1: links
	0: zentriert
	1: rechts
<Angle>	Drehung
<Size>	Schriftgröße in Inch
<Text>	Beschriftung

Beispiel 6

In diesem Beispiel soll für den Jahresgang der Oberflächentemperatur aus dem COADS Datensatz bei 60°N und 30°W jeweils der Winter- und Sommer-Wert an die Kurve geschrieben werden:

```
yes? use coads_climatology
yes? plot/vlim=6:12:.5 sst[y=60N,x=30W]
yes? label `_T[gt=time,t=16-APR]`,6.7,0,0,.12,@C004 `sst[y=60N,x=30W,l=4],prec=-2`
 !-> PPL %LABEL 2557.455,6.7,0,0,.12,@C004 6.97
yes? label `_T[gt=time,t=16-AUG]`,11.2,0,0,.12,@C002 `sst[y=60N,x=30W,l=8],prec=-2`
 !-> PPL %LABEL 5479.395,11.2,0,0,.12,@C002 10.95
```

Im `label`-Kommando wurde die X-Koordinate durch einen eingebetteten Ausdruck angegeben: `` `_T[gt=time,t=16-APR]` `` liefert den tatsächlichen Wert der horizontalen T-Achse (und nicht den Text "APR"). Die Beschriftung für den Winter wird zentriert (0), nicht gedreht (0),mit der Schriftgröße 0.12 Inch und in blau (`@C004`) ausgegeben; der Sommer-Wert entsprechend in Rot:



SEA SURFACE TEMPERATURE (Deg C)

7 Ausgeben und Einlesen von Daten

7.1 Werte ausgeben und abspeichern

Das Kommando zum Ausgeben von Werten ist allgemein:

```
list/<qualifier> <Ausdruck1> [,<Ausdruck2> [,...]]
```

<AusdruckN> kann dabei ein Variablen-Name aus einem Datensatz, eine selbst definierte Variable oder ein arithmetischer Ausdruck sein. Ohne weitere Angaben listet Ferret die Werte des Ausdrucks im Standardformat und der Übersichtlichkeit wegen gestaucht (nicht jede Spalte wird dargestellt) auf dem Bilrdschirm. Allgemeine /<qualifier> sind in der Tabelle 22 aufgeführt. Das Beispiel 7 zeigt ein mögliches Listing der sst-Werte aus dem COADS-Datensatz für Januar zwischen 10N und 10S mit einer Zeilenlänge von 100 Zeichen.

Kommando-Qualifier	Erläuterung
/I=/J=/K=/L=/M=/N=	Index-Region, für die die Ausdrücke ausgewertet werden sollen
/X=/Y=/Z=/T=/E=/F=	Achsen-Region, für die die Ausdrücke ausgewertet werden sollen
/D=	Datensatz, der für die Auswertung genutzt werden soll
/ [NO] HEAD	Gibt einen Header mit zus. Angaben über die Variablen und Gitter aus oder nicht
/NOROWLAB	Unterdrückt die Zeilenbeschriftung zu Beginn jeder Zeile.
/PRECISION=<n>	Gibt die Zahl der Ziffern pro Wert ans (Vor- und Nachkommastellen zusammen). <n> > 0
/WIDTH=<n>	Maximale Zeilenlänge
/SINGLY	Bei mehreren Ausdrücken als Argument wird jeder einzeln ausgegeben (inkl. jeweiligem Header), als wären einzelne Listing-Kommandos gegeben worden.
/FORMAT=<Format>	Legt das zu verwendende Format fest (vorrangig zum Schreiben von Dateien)
/FILE="<Datei>"	Ausgabe in eine Datei mit Namen "<Datei>".

Tabelle 22: Allgemeine Kommando-Qualifier für das Kommando list.

Beispiel 7

```
yes? use coads_climatology
yes? list/y=10S:10N/L=1/width=100 sst
             VARIABLE : SEA SURFACE TEMPERATURE (Deg C)
             FILENAME : coads_climatology.cdf
             FILEPATH : /usr/local/ferret/fer_dsets/data/
             SUBSET   : 180 by 10 points (LONGITUDE-LATITUDE)
             TIME     : 16-JAN 06:00
     ... listing every  15th point
              21E     51E     81E    111E    141E    171E    159W    129W     99W     69W     39W      9W
               1      16      31      46      61      76      91     106     121     136     151     166
 9N   / 50:    ....   25.84   26.99   26.07   27.88   27.77   27.27   26.62   27.12    ....   25.61    ....
 7N   / 49:    ....   26.04   27.22   26.47   28.30   28.49   27.47   26.51   27.08    ....   26.64    ....
 5N   / 48:    ....   26.23   27.52   27.06   28.50   28.82   27.51   26.72   26.67    ....   27.42   28.18
 3N   / 47:    ....   26.28   27.99   27.29   28.87   27.92   27.30   25.61   26.08    ....   27.52   28.01
 1N   / 46:    ....   26.62   28.11   26.52   28.91   28.64   26.62   25.08   24.90    ....   27.48   27.40
 1S   / 45:    ....   27.23   28.22    ....   29.04   29.00   27.04   25.13   24.05    ....   27.52   26.70
 3S   / 44:    ....   27.52   28.09   28.09   28.27   29.12   27.32   25.71   24.16    ....   27.50   26.37
 5S   / 43:    ....   28.22   28.06   28.29    ....   28.98   28.14   25.76   24.34    ....   27.43   26.13
 7S   / 42:    ....   28.44   28.10   28.25   29.55   29.14   28.51   25.95   24.16    ....    ....   25.63
 9S   / 41:    ....   28.23   28.28   28.71   28.79   29.32   28.56   26.54   24.15    ....    ....   24.96
```

Formatierte Ausgabe

Es stehen im Wesentlichen drei Formate zum Erzeugen von neuen Dateien zur Verfügung:

- CDF (NetCDF)
- CSV
- Fortran-Format

CDF Zum Erzeugen einer NetCDF-Datei (Erweiterung *.cdf oder *.nc) kann der Alias save verwendet werden:

```
list/format=CDF /<CDF-Qualifier> <Ausdruck1> [,<Ausdruck2> [,...]]
save
```

Sollte die Datei, deren Name mit `/file="<Datei>"` festgelegt wird, noch nicht existieren, wird sie angelegt und das Ergebnis des Ausdruck1 (bzw. aller Ausdruck$n$) mit samt der Gitterinformationen hineingeschrieben. Soll eine vorhandene Datei überschrieben werden, kann der Qualifier `/clobber` verwendet werden. Sollen neue Variablen einer Datei hinzugefügt, oder zusätzliche Zeitschritte angefügt werden, muss stattdessen `/append` verwendet werden. Darüber hinaus stehen zusätzliche `<CDF-Qualifier>` zur Verfügung (siehe Tabelle 23), mit deren Hilfe die Struktur weiter spezifiziert werden kann.

Um beispielsweise die Januar-SST von COADS separat abzuspeichern genügt:

```
yes? use coads_climatology
yes? save/file="coads_sst_Januar.nc"/L=1 sst
```

CDF Qualifier	Erläuterung
`/FILE="<Datei>"`	Legt den Dateinamen fest
`/ILIMITS=/JLIMITS=/KLIMITS=/LLIMITS=/MLIMITS=/NLIMITS= <Min>:<Max>`	Legt beim ersten Schreiben die Anzahl der Elemente auf der entsprechenden Achse fest, die letztendlich in der Datei enthalten sein wird.
`/XLIMITS=/YLIMITS=/ZLIMITS=/TLIMITS=/ELIMITS=/FLIMITS= <Min>:<Max>`	Legt beim ersten Schreiben den Achsenbereich auf der entsprechenden Achse fest, den letztendlich die Datei enthalten wird.
`/clobber`	Überschreibt eine bereits vorhandene Datei
`/append`	Fügt Daten einer vorhandenen Datei hinzu.
`/ncformat`	Legt die NetCDF-Version fest: `classic` (oder `3`), `netcdf4` (oder `4`), `64bit_offset`
`/deflate`	Schaltet bei netcdf4 die HDF5-Datenkompression ein.
`/keep_axisnames`	Behält die ursprünglichen Achsen-Namen bei, auch wenn der Ausdruck eine Unterregion oder eine Transformation verwendet.
`/title="<Titel>"`	Definiert einen Titel in den globale Attributen der NetCDF-Datei.
`/<WeitereQualifier>`	Für mehr Qualifier siehe commands-reference[15]

Tabelle 23: Kommando-Qualifier für das Kommando `list/format=CDF`.

[15] http://ferret.pmel.noaa.gov/Ferret/documentation/users-guide/commands-reference/LIST

CSV Bei diesem Datenformat werden die Werte als ASCII-Text mit einem Trennzeichen dazwischen ausgegeben. Das Trennzeichen kann entweder ein Komma sein oder ein Tabulator-Zeichen. Die entsprechenden `list`-Kommandos lauten:

list/format=comma/<CSV-Qualifier> <Ausdruck1> [**,<Ausdruck2>** [**,...**]]

list/format=tab/<CSV-Qualifier> <Ausdruck1> [**,<Ausdruck2>** [**,...**]]

Beispiel 8

```
yes? use coads_climatology
yes? list/format=comma/file="coads_sst_Januar.csv"/y=10S:10N/x=50W:20W/l=1/clob/width=100 sst
 LISTing to file coads_sst_Januar.csv
yes? sp cat coads_sst_Januar.csv
             VARIABLE : SEA SURFACE TEMPERATURE (Deg C)
             FILENAME : coads_climatology.cdf
             FILEPATH : /usr/local/ferret/fer_dsets/data/
             BAD FLAG : -1.E+34
             SUBSET   : 15 by 10 points (LONGITUDE-LATITUDE)
             TIME     : 16-JAN 06:00
   49W  ,47W  ,45W  ,43W  ,41W  ,39W  ,37W  ,35W  ,33W  ,31W  ,29W  ,27W  ,25W  ,23W  ,21W
9S,-1.E+34,-1.E+34,-1.E+34,-1.E+34,-1.E+34,-1.E+34,-1.E
     +34,27.56,27.49,27.23,26.98,27.04,26.64,26.42,26.24
7S,-1.E+34,-1.E+34,-1.E+34,-1.E+34,-1.E+34,-1.E+34,-1.E
     +34,27.59,27.45,27.31,27.16,26.79,26.77,26.78,26.49
5S,-1.E+34,-1.E+34,-1.E+34,-1.E+34,-1.E
     +34,27.43,27.57,27.58,27.22,27.15,26.99,27.08,26.79,26.6,26.42
3S,-1.E+34,-1.E+34,25.93,26.91,27.56,27.5,27.52,27.57,27.18,27.16,27.14,27.1,26.66,26.74,26.54
1S,27.38,27.69,27.72,27.58,27.52,27.52,27.43,27.34,27.25,27.1,27.09,27.04,26.71,26.77,26.91
1N,27.7,27.5,27.53,27.5,27.42,27.48,27.47,27.22,27.19,27.05,27.04,26.91,27.11,27.29,27.06
3N,27.4,27.38,27.44,27.36,27.4,27.52,27.49,27.11,27.12,27.05,27.05,27.16,27.3,27.36,27.33
5N,27.29,27.1,27.12,27.31,27.31,27.42,27.19,27.1,26.88,26.99,26.95,26.92,27.1,27.01,27.36
7N,26.93,26.89,27.02,27,27.08,26.64,26.53,26.65,26.52,26.59,26.5,26.53,26.54,26.81,27.02
9N,26.86,26.68,26.48,26.59,25.98,25.61,26.26,26.28,26.22,25.9,25.94,25.81,25.81,26.21,26.06
```

Fortran-Format Das sogenannte Fortran-Format basiert auf einer Funktionalität der Programmiersprache FORTRAN, mit dessen Hilfe sich die Art und Weise beeinflussen lässt, wie eine Zahl dargestellt werden soll. Obwohl das Format auch Ganzahlige und nicht-numerische Werte berücksichtig, kann Ferret nur reine Fließkomma-Zahlenwerte in diesem Format ausgeben. Das Kommando lautet allgemein:

list/format="(<FortranFormat>)"/<Qualifier> <Ausdruck1> [**,<Ausdruck2>** [**,...**]]

"(<FotranFormat>)" enthält eine Reihe bestimmter Format-Specifier, die das Ausgabeformat pro Spalte definieren. Diese Specifier haben die Form:

[*n*] F*w*.*d*	Fließkomma
[*n*] E*w*.*d*	Exponential
[*n*] G*w*.*d*	Kombiniert F/E

Dabei steht *w* für die Gesamtzahl der Ziffern inkl. Dezimalpunkt und Exponential-Zeichen und *d* für die Nachkommastellen. n ist ein optionaler, ganzzahliger Wiederholungsfaktor, der den nachfolgenden Format-Specifier wiederholt.

Als Beispiel wird eine Variable "`zahl`" definiert und mit verschiedenen Formaten ausgegeben[16]; zunächst mit dem Standard-Format (ohne jede `/format=` Angabe).

[16] Aus gründen der besseren Lesbarkeit werden Leerziechen als "␣" ausgegeben.

Standardformat

```
yes?␣let␣zahl=12345.6789

yes?␣list␣zahl
␣␣␣␣␣␣␣␣␣␣␣␣VARIABLE␣:␣12345.6789
␣␣␣␣␣␣␣␣␣␣12346.
```

Das Standard-Format von Ferret rundet in diesem Fall auf eine ganze Zahl auf.

Floating Point (nFw.d)

```
yes?␣list/format="(F10.4)"␣zahl
12345.6789

yes?␣list/format="(F10.3)"␣zahl
␣12345.679

yes?␣list/format="(F9.4)"␣zahl
*********

yes?␣list/format="(F15.6)"␣zahl
␣␣␣12345.678900
```

Da die Zahl 5 Stellen vor dem Komma und 4 nach dem Komma besitzt, nimmt sie mit dem Dezimalpunkt genau 10 Zeichen (ohne Minus-Zeichen) ein. Die Nachkommastellen können ohne weiteres gekürzt werden. Bei der Ausgabe wird dann entsprechend gerundet. Werden jedoch zu wenig Stellen vor dem Komma vorgesehen, werden Sternchen pro Ziffer (*********) als Fehler ausgegeben. Hat die Zahl weniger Stellen vor dem Komma, als vorgesehen, wird mit Leerzeichen aufgefüllt; Nachkommastellen werden mit Nullen aufgefüllt, um sicherzustellen, dass der Dezimal-Punkt stets an der selben Zeichenspalte pro Zeile steht.

Exponential (nEw.d)

```
yes?␣list/format="(E10.4)"␣zahl
0.1235E+05

yes?␣list/format="(E10.3)"␣zahl
␣0.123E+05

yes?␣list/format="(E10.1)"␣zahl
␣␣␣0.1E+05

yes?␣list/format="(E8.4)"␣zahl
********


yes?␣list/format="(E20.8)"␣zahl
␣␣␣␣␣␣0.12345679E+05
--------------------------------
12345678901234567890
0␣␣␣␣␣␣␣␣1␣␣␣␣␣␣␣␣2
```

Im Exponential-Format verschiebt Ferret sämtliche Stellen hinter das Komma und passt dafür den 10er-Exponent an. *n* und *d* (inkl. Dezimalpunkt und Exponential-Zeichen E plus Vorzeichen) legen fest, wieviele Nachkommastellen erhalten bleiben.

Floating Pont/Exponential kmobiniert (nGw.d)

```
yes?␣list/format="(G10.4)"␣zahl
0.1235E+05

yes?␣list/format="(G10.3)"␣zahl
␣0.123E+05
yes?␣list/format="(G8.4)"␣zahl
********

yes?␣list/format="(G11.4)"␣zahl
␣0.1235E+05

yes?␣list/format="(G20.10)"␣zahl
␣␣␣␣␣12345.67890
```

```
------------------------------
12345678901234567890
0␣␣␣␣␣␣␣␣1␣␣␣␣␣␣␣␣2
```

Das kombinierte Format aus F und E zeigt nahezu das gleiche an, wie das Format E. Erst wenn ausreichend Stellen zur Verfügung stehen, ohne dass gerundet werden muss, wechselt das Format auf F.

Wiederholungszeichen (n)

```
yes?␣list/format="(F12.4)"␣zahl,zahl
␣␣12345.6789
␣␣12345.6789

yes?␣list/format="(F12.4F12.4)"␣zahl,zahl
␣␣12345.6789␣␣12345.6789

yes?␣list/format="(2F12.4)"␣zahl,zahl
␣␣12345.6789␣␣12345.6789
```

Sollen mehrere Zahlen oder ein Vektor pro Zeile ausgegeben werden, müssen auch entsprechend viele Specifier definiert werden. Um die Formatsequenz abzukürzen kann das Widerholungszeichen *n* vor dem Format-Descriptor verwendet werden.

7.2 Direkt-Eingabe von Daten

Selten werden Daten in Ferret von Hand eingegeben und wenn doch, dann meistens als skalare Parameter für Berechnungen (Siehe auch den Abschnitt 7.3 “Einlesen von Datensätzen)”.

Skalar

Ferret	*Matlab (zur Erinnerung)*
`yes? let n = 8`	`>> n = 8`

Zeilenvektor (Tupel entlang X-Achse)

Vektoren werden in Ferret immer entlang X definiert.

`yes? let a = {1,2,,3}`	`>> a = [1 2 NaN 3]`

Zahlenkollonnen können mittels der sog. Pseudo-Variablen erzeugt werden (Siehe auch 2.5 Zugriff auf einzelne Elemente oder Bereiche einer Variablen):

```
yes? let xx = I[i=1:8:2]                    >> x = 1:2:n
yes? let yy = Y[y=1:10:.5]
yes? let zz = Z[z=0:6000:1000]
```

Spaltenvektor (entlang Y-, Z-, T-Achse)

Um Werte in einen Vektor antlang einer anderen Achse als X zu schreiben, muss der X-Vektor mittels der Funktionen `ysequence()`, `zsequence()` oder `tsequence()` auf die Y-, Z- oder T-Achse gebracht werden.

```
yes? let b = ysequence(a)                   >> b = a'
yes? let b = zsequence(a)
yes? let b = tsequence(a)
```

Beispiel 9

```
yes? let a={1,2,,3}
yes? let b = ysequence(a)

yes? sh gr b
    GRID YABSTRACT
 name       axis              # pts   start                end
 normal     X
 ABSTRACT   Y              9999999 r   1                     9999999
 normal     Z
 normal     T

yes? list b
             VARIABLE : YSEQUENCE(A)
             SUBSET   : 4 points (Y)
 1   / 1:  1.000
 2   / 2:  2.000
 3   / 3:   ....
 4   / 4:  3.000


yes? let c = zsequence(a)
yes? sh gr c
    GRID ZABSTRACT
 name       axis              # pts   start                end
 normal     X
 normal     Y
 ABSTRACT   Z              9999999 r   1                     9999999
 normal     T
yes? list c
             VARIABLE : ZSEQUENCE(A)
             SUBSET   : 4 points (Z)
 1   / 1:  1.000
 2   / 2:  2.000
 3   / 3:   ....
 4   / 4:  3.000
```

Matrizen

Matrizen bzw. Arrays sind in Ferret als Direkteingabe schwierig zu relisieren und werden besser als ASCII-Daten eingelesen.

7.3 Einlesen von Datensätzen

Ferret unterscheidet im Wesentlichen drei Datentypen: NetCDF, ASCII und Binär.

NetCDF

Im Abschnitt 2.2 wurde schon ein Beispiel für die Benutzung des Kommandos `use` zum Einlesen von NetCDF-Daten (Dateinamenserweiterung *.cdf oder *.nc) gezeigt. Dabei handelte es sich um ein Alias für eine bestimmte Form des Kommandos `set data_set`:

$$\underbrace{\texttt{set data\_set/format=CDF}}_{\texttt{use}} \; [\text{"}<\text{path}>/] \; \texttt{<Dateiname>} \; [\text{"}]$$

Enthält der `<Dateiname>` auch Pfadangaben `<path>` (sowas wie: `/path/to/data`) würde Ferret versuchen, die Pfadanteile zwischen den Schrägstrichen, als Kommando-Qualifier zu interpretieren. Um dies zu verhindern, muss <Dateiname> in Anführungsstriche (“”) gesetzt werden.

ASCII

ASCII-Daten können mit dem Kommando-Alias `file` eingelesen werden[17].

```
set data_set/EZ /format=<ASCII-Format>/<Qualifier> " <Dateiname> "
     file
```

Für /format=<ASCII-Format> stehen in diesem Fall zusätzlich zur Auswahl:

- `/format=FREE`

  Datei enthält Trennzeichen (kann Komma, Leerzeichen oder Tabulatorzeichen sein); oft auch CSV-Format genannt (von 'comma separated values'). Alle Werte müssen aber numerisch sein.

- `/format=DELIMITED/delimiter="<TrennZeichen>"`

  Datei enthält Trennzeichen (kann bestimmt werden; ohne `/delimiter` Angaben: Komma oder Tabulator-Zeichen); kann ebenfalls als CSV verstanden werden. Werte können numerisch oder Text sein.

- `/format="( <FortranFormat> )"`

  Werte wurden in einem `<Fortran-Format>` abgespeichert

FREE bzw. DELIMITED (CSV) Beide Datentypen lassen sich in Ferret am bequemsten mit dem Alias `column` einladen, also als ASCII-Datensatz(`/EZ`) vom Format "`DELIMITED`" mit einem typischen Trennzeichen (Komma oder Tabulator):

```
set data_set/EZ/format=DELIMITED /<Qualifier> " <Dateiname> "
               column
```

Es stehen einige `/<Qualifier>` zur Verfügung (siehe Tabelle 24), mit deren Hilfe die Struktur des ASCII-Datensatzes weiter spezifiziert werden kann. So können alternative Trennzeichen festgelegt werden (`/delimiter=`), der Datei-Kopf kann übersprungen werden (`/skip`) bzw. die Spalten können eigenen Variablen-Namen zugeordnet und Spalten übersprungen (`/variables=`) werden oder die Daten gleich mit einem bereits definiertem Gitter verknüpft werden (`/grid`, `/order`).

[17] Anders als beim Alias `use` ist der `/format`-Qualifier nicht teil des Alias `file`. Dieses wird benötigt, um weitere Spezifikationen vorzunehmen.

Kommando-Qualifier	Erläuterung
/delimiter="<TrennZeichen>"	Optionale Liste möglicher Trennzeichens. Wenn weggelassen dann wird Standardwert gesetzt: "\t,\," (Tab und Komma) Mögliche Sonderzeichen: \b (Leerzeichen), \t (Tab), \n (Zeilenumbruch), \nnn (3-stelliger Index der ASCII-Tabelle)
/skip=<n>	Überspringt die ersten <n> Zeilen (z.B. Überschriften/Dateikopf)
/variables="<var1> [,<var2> [,...]]"	Bestimmt einen Variablenname pro Spalte. Ein Minus "-" signalisiert, dass diese Spalte/Variable ausgelassen werden kann
/columns=<n>	Anzahl der Spalten; jede Spalte ist eine andere Variable.
/type=<VarTyp>	Der Variablentyp (NUMERIC, TEXT, LATITUDE, LONGITUDE, DATE (mm/dd/yy, mm/dd/yyyy, yyyy-mm-dd, yyyymmdd in Tagen seit 01-JAN-1900), EURODATE (dd/mm/yy, dd/mm/yyy, yyyy-mm-dd) oder TIME (hh:mm, hh:mm:ss.s) oder "–" (auslassen)
/grid=<Grid>	Verknüpft die Daten direkt mit einem Gitter.
/order=<Permutation>	<Permutation> ist eine Folge von Achsen-Bezeichnungen (XYZT), die die Reihenfolge festlegen, in der Achsen eingelesen werden.

Tabelle 24: Kommando-Qualifier für Kommando `set data_set`.

Da es eine Vielzahl unterschiedlichster CSV-Strukturen gibt, werden hier nur einige typische Beispiele vorgestellt (der Inhalt der jeweiligen Dateien ist im Anhang G aufgeführt):

1. Eine Spalte, keine Überschrift: data1.csv

Diese Datei enthält nur eine Spalte, keinerlei Überschriften und die Werte sind alle numerisch. Beim Einladen mit `file` wird auch nur eine Variable gefunden und von Ferret automatisch "`v1`" genannt. Die Angabe der Elementezahl für die erste Dimension ist zunächst falsch (`1:20480`). Erst nach einer tatsächlichen Auswertung mit `list` merkt Ferret, wieviele Einträge tatsächlich vorhanden sind (12), weil erst dann die Datei vollständig gelesen werden.

```
yes? file data1.csv
yes? sh da
     currently SET data sets:
    1> ./data1.csv  (default)
 name     title                I          J          K          L          M          N
 V1       V1                   1:20480    ...        ...        ...        ...        ...

yes? list v1
             VARIABLE : V1
             FILENAME : data1.csv
             SUBSET   : 12 points (X)
 1    /  1:   7.40
 2    /  2:   7.08
 3    /  3:   7.02
 4    /  4:   6.97
 5    /  5:   7.46
 6    /  6:   8.95
 7    /  7:  10.44
 8    /  8:  10.95
 9    /  9:  10.42
 10   / 10:   9.09
 11   / 11:   8.18
 12   / 12:   7.47

yes? sh da
     currently SET data sets:
    1> ./data1.csv  (default)
 name     title                I          J          K          L          M          N
 V1       V1                   1:12       ...        ...        ...        ...        ...
```

2. Zwei Spalten, Komma getrennt, Überschrift: data2.csv

Diese Datei enthält zwei Spalten und Überschriften in der ersten Zeile. Über den Qualifier `/variables` landet die erste Spalte in der Variablen "Monat" und die zweite in "SST".

```
yes? column/var=Monat,SST data2.csv
yes? sh da
     currently SET data sets:
    1> ./data2.csv  (default)
 name     title                          I         J         K         L         M         N
 MONAT    MONAT                          1:20480   ...       ...       ...       ...       ...
 SST      SST                            1:20480   ...       ...       ...       ...       ...

yes? list monat,sst
             DATA SET: ./data2.csv
             X: 0.5 to 13.5
 Column  1: MONAT
 Column  2: SST
                MONAT     SST
1    /  1: "Monat"     "SST"
2    /  2: "Januar"    "7.396"
3    /  3: "Februar"   "7.079"
4    /  4: "März"     "7.019"
5    /  5: "April"     "6.969"
6    /  6: "Mai"       "7.462"
7    /  7: "Juni"      "8.950"
8    /  8: "Juli"      "10.443"
9    /  9: "August"    "10.946"
10   / 10: "September" "10.416"
11   / 11: "Oktober"   "9.088"
12   / 12: "November"  "8.177"
13   / 13: "Dezember"  "7.474"

yes? say `sst,r=dtype`
 !-> MESSAGE/CONTINUE CHAR
CHAR
```

Allerdings wurden, wegen der Überschrift, die Einträge in der zweiten Spalte als Text (Character) eingelesen. Es gibt zwei Wege das zu vermeiden:

1. Die erste Zeile wird mit `/skip=1` übersprungen, oder
2. der Variablentyp wird für die Spalten explizit angegeben (`/typ="TEXT,NUMERIC"`)

Die zweite Variante hat zudem den Vorteil, dass Fehlerwerte, wenn sie nicht durch numerische Werte repräsentiert werden (etwa `-9999.99`) sondern durch Text (z.B. `NaN` oder `BAD`), von Ferret richtig interpretiert werden.

```
yes? !  ==1.==

yes? column/var=Monat,SST/skip=1 data2.csv
yes? list monat,sst
             DATA SET: ./data2.csv
             X: 0.5 to 12.5
 Column  1: MONAT
 Column  2: SST
                MONAT    SST
1    /  1: "Januar"      7.40
2    /  2: "Februar"     7.08
3    /  3: "März"       7.02
4    /  4: "April"       6.97
5    /  5: "Mai"         7.46
6    /  6: "Juni"        8.95
7    /  7: "Juli"       10.44
8    /  8: "August"     10.95
9    /  9: "September"  10.42
10   / 10: "Oktober"     9.09
11   / 11: "November"    8.18
12   / 12: "Dezember"    7.47

yes? say `sst,r=dtype`
 !-> MESSAGE/CONTINUE FLOAT
FLOAT


yes? !  ==2.==

yes? column/var=Monat,SST/type="TEXT,NUMERIC" data2.csv
yes? sh da
```

```
     currently SET data sets:
    1> ./data2.csv  (default)
 name     title                          I         J         K         L         M         N
 MONAT    MONAT                          1:20480   ...       ...       ...       ...       ...
 SST      SST                            1:20480   ...       ...       ...       ...       ...

yes? list monat,sst
             DATA SET: ./data2.csv
             X: 0.5 to 13.5
 Column  1: MONAT
 Column  2: SST
                MONAT    SST
1     /  1: "Monat"       ....
2     /  2: "Januar"      7.40
3     /  3: "Februar"     7.08
4     /  4: "März"       7.02
5     /  5: "April"       6.97
6     /  6: "Mai"         7.46
7     /  7: "Juni"        8.95
8     /  8: "Juli"       10.44
9     /  9: "August"     10.95
10    / 10: "September"  10.42
11    / 11: "Oktober"     9.09
12    / 12: "November"    8.18
13    / 13: "Dezember"    7.47

yes? say `sst,r=dtype`
 !-> MESSAGE/CONTINUE FLOAT
FLOAT
```

3. Zwei Spalten, durch Leerzeichen getrennt, Überschrift: data2a.csv

Diese Datei ist data2.csv sehr ähnlich, besitzt aber Leerzeichen als Trennzeichen, welche von `column` so nicht erkannt werden. Alles landet in der Text-Variablen `MONAT`.

```
yes? column/var=Monat,SST/type="TEXT,NUMERIC" data2a.csv
yes? sh da
     currently SET data sets:
    1> ./data2a.csv  (default)
 name     title                          I         J         K         L         M         N
 MONAT    MONAT                          1:20480   ...       ...       ...       ...       ...
 SST      SST                            1:20480   ...       ...       ...       ...       ...

yes? list MONAT,SST
             DATA SET: ./data2a.csv
             X: 0.5 to 13.5
 Column  1: MONAT
 Column  2: SST
                      MONAT  SST
1     /  1: "Monat  SST"       ....
2     /  2: "Januar   7.396"   ....
3     /  3: "Februar   7.079"  ....
4     /  4: "März   7.019"    ....
5     /  5: "April   6.969"    ....
6     /  6: "Mai   7.462"      ....
7     /  7: "Juni   8.950"     ....
8     /  8: "Juli  10.443"     ....
9     /  9: "August  10.946"   ....
10    / 10: "September  10.416"....
11    / 11: "Oktober   9.088"  ....
12    / 12: "November   8.177" ....
13    / 13: "Dezember   7.474" ....
```

Um diese Datei wieder korrekt einzulesen, wird das Trennzeichen mit `/delimiter="\b"` auf Leerzeichen (\b='Blank') gesetzt.

```
yes? column/delimiter="\b"/var=Monat,SST/type="TEXT,NUMERIC" data2a.csv
yes? sh da
     currently SET data sets:
    1> ./data2a.csv  (default)
 name     title                          I         J         K         L         M         N
 MONAT    MONAT                          1:20480   ...       ...       ...       ...       ...
 SST      SST                            1:20480   ...       ...       ...       ...       ...

yes? list MONAT,SST
             DATA SET: ./data2a.csv
             X: 0.5 to 13.5
 Column  1: MONAT
 Column  2: SST
                MONAT    SST
```

```
1    /  1: "Monat"       ....
2    /  2: "Januar"      7.40
3    /  3: "Februar"     7.08
4    /  4: "März"       7.02
5    /  5: "April"       6.97
6    /  6: "Mai"         7.46
7    /  7: "Juni"        8.95
8    /  8: "Juli"       10.44
9    /  9: "August"     10.95
10   / 10: "September"  10.42
11   / 11: "Oktober"     9.09
12   / 12: "November"    8.18
13   / 13: "Dezember"    7.47
```

4. Drei Spalten, Datum, Überschrift: data3.csv

In diesem Datensatz stehen drei Spalten (das Datum, sst und airt). Leider entspricht das Datums-Format in der Datei keinem Typ in Ferret, daher muss es wieder als Text eingelesen werden. In diesem Fall wird aber ein zuvor definiertes Gitter mit diesem Datensatz verknüpft. Hierzu werden die klimatolgischen Zeitachsen von Ferret geladen *(climatological_ axes*, siehe auch 5.1). Aus der Zeitachse 'month_gregorian' wird dann mit `define grid` ein neues Gitter namens `myg1` erzugt und dieses mit dem `/grid`-Qualifier im `column`-Kommando mitgegeben. Dadurch nimmt Ferret sofort an, dass es 12 Einträge im Datensatz vorfindet.

```
yes? use climatological_axes
 *** NOTE: regarding /usr/local/PyFerret_1.0.2/go/climatological_axes.cdf ...
 *** NOTE: Climatological axes SEASONAL_REG, MONTH_REG, MONTH_IRREG, MONTH_GREGORIAN, MONTH_NOLEAP
      , MONTH_360_DAY, MONTH_ALL_LEAP  defined
yes? ca da climatological_axes
yes? define grid/t=month_gregorian myg1
yes? sh gr myg1
    GRID MYG1
 name       axis              # pts   start                end
 normal     X
 normal     Y
 normal     Z
 MONTH_GREGORIAN TIME          12mi   16-JAN 12:00         15-DEC 17:49
 normal     E
 normal     F
yes? ca da/a
yes? column/delimiter="\b"/var="Datum,SST,AIRT"/type="TEXT,NUMERIC,NUMERIC"/skip=1/g=myg1 data3.
     csv
yes? sh da
     currently SET data sets:
    1> ./data3.csv  (default)
 name     title                              I         J         K         L         M         N
 DATUM    DATUM                             ...       ...       ...       1:12      ...       ...
 SST      SST                               ...       ...       ...       1:12      ...       ...
 AIRT     AIRT                              ...       ...       ...       1:12      ...       ...

yes? sh gr sst
    GRID MYG1
 name       axis              # pts   start                end
 normal     X
 normal     Y
 normal     Z
 MONTH_GREGORIAN TIME          12mi   16-JAN 12:00         15-DEC 17:49
 normal     E
 normal     F
yes? plot sst
yes? list datum,sst,airt
             DATA SET: ./data3.csv
             TIME: 01-JAN 00:00 to 31-DEC 05:49
 Column  1: DATUM
 Column  2: SST
 Column  3: AIRT
                       DATUM    SST   AIRT
16-JAN 12      /  1: "16-JAN"   7.40   4.14
15-FEB 02      /  2: "15-FEB"   7.08   4.24
15-MAR 17      /  3: "17-MAR"   7.02   4.33
15-APR 05      /  4: "16-APR"   6.97   5.96
15-MAY 17      /  5: "16-MAY"   7.46   6.99
15-JUN 05      /  6: "16-JUN"   8.95   8.89
15-JUL 17      /  7: "16-JUL"  10.44  10.19
15-AUG 17      /  8: "16-AUG"  10.95  10.73
15-SEP 05      /  9: "15-SEP"  10.42   9.85
15-OCT 17      / 10: "16-OCT"   9.09   7.78
15-NOV 05      / 11: "15-NOV"   8.18   5.96
15-DEC 17      / 12: "16-DEC"   7.47   5.05
```

5. Drei Spalten, Stundenangabe, Überschrift: data4.csv

Im letzten Beispiel soll eine Variante des data3.csv Datensatzes eingelesen werden, bei dem in der ersten Spalte jetzt eine rein numerische Zeitangabe steckt, nämlich die Stunden seit 01-Jan-0000. D.h. der Datensatz wird eingelesen, aus den Werten der ersten Spalte eine Zeitachse generiert und die Variablen damit verknüpft:

```
yes? column/delimiter="\b"/var="TIME,SST,AIRT"/type="NUMERIC,NUMERIC,NUMERIC"/skip=1 data4.csv
yes? sh da
     currently SET data sets:
    1> ./data4.csv  (default)
 name     title                          I         J         K         L         M         N
 TIME     TIME                        1:20480   ...       ...       ...       ...       ...
 SST      SST                         1:20480   ...       ...       ...       ...       ...
 AIRT     AIRT                        1:20480   ...       ...       ...       ...       ...

yes? list time
             VARIABLE : TIME
             FILENAME : data4.csv
             SUBSET   : 12 points (X)
 1    /  1:   366.
 2    /  2:  1096.
 3    /  3:  1827.
 4    /  4:  2557.
 5    /  5:  3288.
 6    /  6:  4018.
 7    /  7:  4749.
 8    /  8:  5479.
 9    /  9:  6210.
 10   / 10:  6940.
 11   / 11:  7671.
 12   / 12:  8401.

yes? def ax/t/cal=gregorian/t0=01-JAN-0000/mod tax=time
yes? sh ax/t tax
 name       axis              # pts   start                end
 TAX       TIME               12mi   16-JAN 06:00         16-DEC 01:00
T0 = 01-JAN-0000
   Axis span (to cell edges) = 8765 (modulo length = 8765.82)

       L     T                   TBOX      TBOXLO                TSTEP (HOURS)
       1>  16-JAN 06:00:00       730        01-JAN 01:00:00        366
       2>  15-FEB 16:00:00       730.5      31-JAN 11:00:00        1096
       3>  17-MAR 03:00:00       730.5      01-MAR 21:30:00        1827
       4>  16-APR 13:00:00       730.5      01-APR 08:00:00        2557
       5>  17-MAY 00:00:00       730.5      01-MAY 18:30:00        3288
       6>  16-JUN 10:00:00       730.5      01-JUN 05:00:00        4018
       7>  16-JUL 21:00:00       730.5      01-JUL 15:30:00        4749
       8>  16-AUG 07:00:00       730.5      01-AUG 02:00:00        5479
       9>  15-SEP 18:00:00       730.5      31-AUG 12:30:00        6210
      10>  16-OCT 04:00:00       730.5      30-SEP 23:00:00        6940
      11>  15-NOV 15:00:00       730.5      31-OCT 09:30:00        7671
      12>  16-DEC 01:00:00       730        30-NOV 20:00:00        8401
yes? define grid/t=tax myg2
```

Mit dem neuen Gittter wird die Datei nochmals eingelesen und die erste Spalte übersprungen, da die Zeitinformation bereits als Achse im Speicher ist:

```
yes? column/delimiter="\b"/var="-,SST,AIRT"/type="-,NUMERIC,NUMERIC"/skip=1/g=myg2 data4.csv
yes? sh da
     currently SET data sets:
    1> ./data4.csv  (default)
 name     title                          I         J         K         L         M         N
 SST      SST                         ...       ...       ...       1:12      ...       ...
 AIRT     AIRT                        ...       ...       ...       1:12      ...       ...

yes? sh gr sst
    GRID MYG2
 name       axis              # pts   start                end
 normal    X
 normal    Y
 normal    Z
 TAX       TIME               12mi   16-JAN 06:00         16-DEC 01:00
 normal    E
 normal    F
yes? sh ax/t tax
 name       axis              # pts   start                end
 TAX       TIME               12mi   16-JAN 06:00         16-DEC 01:00
T0 = 01-JAN-0000
   Axis span (to cell edges) = 8765 (modulo length = 8765.82)

       L     T                   TBOX      TBOXLO                TSTEP (HOURS)
```

```
      1>  16-JAN 06:00:00       730        01-JAN 01:00:00        366
      2>  15-FEB 16:00:00       730.5      31-JAN 11:00:00        1096
      3>  17-MAR 03:00:00       730.5      01-MAR 21:30:00        1827
      4>  16-APR 13:00:00       730.5      01-APR 08:00:00        2557
      5>  17-MAY 00:00:00       730.5      01-MAY 18:30:00        3288
      6>  16-JUN 10:00:00       730.5      01-JUN 05:00:00        4018
      7>  16-JUL 21:00:00       730.5      01-JUL 15:30:00        4749
      8>  16-AUG 07:00:00       730.5      01-AUG 02:00:00        5479
      9>  15-SEP 18:00:00       730.5      31-AUG 12:30:00        6210
     10>  16-OCT 04:00:00       730.5      30-SEP 23:00:00        6940
     11>  15-NOV 15:00:00       730.5      31-OCT 09:30:00        7671
     12>  16-DEC 01:00:00       730        30-NOV 20:00:00        8401
```

Fortran-Format Das Einlesen von Fortran-formatierten ASCII-Daten folgt den selben Formatierungsregeln wie das Ausgeben/Wegschreiben (siehe Abschnitt Fortran-Format auf Seite 61). Das Kommando lautet entsprechend:

set data_set/format="(<FortranFormat>)"/<Qualifier> [”] **<Dateiname>** [”]

Da zum Beispiel die Datei `data4.csv` einen gleichförmigen Spaltenaufbau hat, kann diese auch mit dem Fortran-Format eingelesen werden:

1.Spalte: 7 Zeichen mit Dezimalpunkt, keine Nachkommastellen

2. Spalte: 7 Zeichen inkl. Dezimalpunkt mit 3 Nachkommastellen

3. Spalte: 8 Zeichen inkl. Dezimalpunkt mit 3 Nachkommstellen

Das Einlesen erfolgt daher (mit Überspringen der Überschriften und mit drei selbst vergebenen Variablennamen) folgendermaßen:

```
yes? se da/format="(F7.0F7.3G8.3)"/var=time,sst,airt/skip=1 data4.csv
yes? list/norowlab time,sst,airt
            DATA SET: ./data4.csv
            X: 0.5 to 12.5
 Column  1: TIME
 Column  2: SST
 Column  3: AIRT
  TIME     SST   AIRT
   366.   7.40   4.14
  1096.   7.08   4.24
  1827.   7.02   4.33
  2557.   6.97   5.96
  3288.   7.46   6.99
  4018.   8.95   8.89
  4749.  10.44  10.19
  5479.  10.95  10.73
  6210.  10.42   9.85
  6940.   9.09   7.78
  7671.   8.18   5.96
  8401.   7.47   5.05
```

Binär

Binäre Formate sind sehr speziell und ohne zusätzliche Informationen sehr schwer einzulesen.

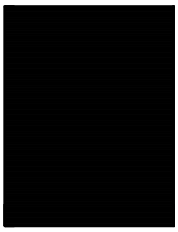
A Kommando Referenz

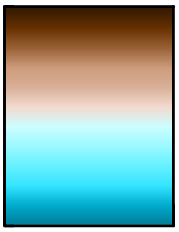
Kommando	Erweiterung	Qualifier	Argument	Erläuterung
ALIAS				*-> DEFINE ALIAS*
CANCEL	ALIAS		<ALIAS_NAME>	
	ATTRIBUTE	/OUTPUT /DATASET	<VARNAME>.<ATTNAME>	
	AXIS	/MODULO /DEPTH /ALL /STRIDE	<AXIS_NAME>	
	DATA_SET	/ALL /NOERROR	[<DATASET> [,<DATASET> [,...]]]	
	EXPRESSION			
	GRID		<GRID_NAME>	
	LIST	/ALL /APPEND /FILE /FORMAT /HEADING /OUTTYPE /PRECISION		
	MEMORY	/ALL /PERMANENT /TEMPORARY		
	MODE		<MODE> siehe Mode-Tabelle	
	MOVIE			
	NCCACHE			
	REDIRECT			
	REGION	/I/J/K/L/M/N /X/Y/Z/T/E/F /ALL	[<REGION_NAME>]	
	SYMBOL	/ALL	[<SYMBOL>]	
	VARIABLE	/ALL /DATASET	[<VAR_NAME>]	
	VIEWPORT		<VIEW_NAME>	
	WINDOW	/ALL	[<WINDOWS_NUMBER>]	
	PYVAR			
CONTOUR		/I/J/K/L/M/N /X/Y/Z/T/E/F /D /FILL /FRAME /KEY /LEVELS /LINE /NOAXIS /NOKEY /NOLABEL /OVERLAY /PALETTE /PATTERN /SIZE /SPACING /SIGDIG /PEN /SET_UP /TITLE /COLOR /TRANSPOSE /HLIMITS /VLIMITS /AXES /HGRATICULE /VGRATICULE /GRATICULE /MODULO /OPACITY	<EXPRESSION> oder <EXPRESSION>, <XCOORD>, <YCOORD>	*Plottet ein Feld mit Kontour-Linien. flächen gleicher Werte können mit einer Farbe und/oder mit schraffuren gefüllt werden.*
DEFINE	ALIAS		<NAME> <COMMAND>	*Definiert Kommando-Ersetzung (Alias)*
	ATTRIBUTE	/DATASET /TYPE /OUTPUT	<VARNAME>.<ATTNAME> = <EXPRESSION>	*Legt ein NetCDF Attribut für einer Variablen oder global an.*
	AXIS	/X/Y/Z/T/E/F /DEPTH /FILE /FROM_DATA /MODULO /NAME /NPOINTS /TO /UNITS /EDGES /CALENDAR /BOUNDS	<AXIS_NAME> oder <AXIS_NAME> = <EXPRESSION>	*Definiert eine Achse.*
	GRID	/X/Y/Z/T/E/F /FILE /LIKE	GRID_NAME	*Definiert aus Achsen ein Gitter.*
	REGION	/I/J/K/L/M/N /X/Y/Z/T/E/F /DI/DJ/DK/DL/DM/DN /DX/DY/DZ/DT/DE/DF /DEFAULT	[<REGION_NAME>] first 4 characters are recognized	*Definiert eine benannte Region.*
	SYMBOL	/D /QUIET /TITLE /UNITS /BAD=	<SYMBOLNAME> = <STRING>	*Legt eine Text-Constante (Symbol) fest.*
	VARIABLE	/D /QUIET /TITLE /UNITS /BAD=	<VAR_NAME> = <EXPRESSION>	*Definiert eine neue Variable.*
	VIEWPORT	/CLIP /ORIGIN /SIZE /TEXT /XLIMITS /YLIMITS/AXES	<VIEW_NAME>	*Definiert einen benannten Viewport*
	DATA	/AGGREGATE /E /TITLE /QUIET/HIDE	<ENSEMBLE_NAME> = <COMMA_LIST_OF DATASETS>	*Fasst mehrere gleichförmige Daten zum Ensemble zusammen.*
	PYFUNC	/NAME=alias		*Definiert eine Python Funktion.*
ELIF				
ELSE				
ENDIF				
EXIT		/LOOP /SCRIPT /PROMPT /PROGRAM /COMMAND /CYCLE /TOPYTHON		*Beenedet ferret, ein Skript, oder eine Schleife.*
FILE				*-> SET DATA/EZ*
FILL				*-> CONTOUR/FILL*
FRAME		/FORMAT= /FILE= /TRANSPAR		*Speichert den aktuellen Grafikfensterinhalt als Bild ab.*
GO		/HELP	<FILE_NAME> [<ARG> [<ARG> [...]]]	*Führt ein benanntes Skript aus.*

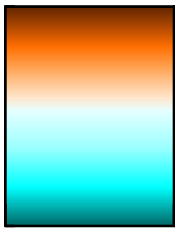
HELP				
IF			(<CONDITION>) THEN <COMMAND> [ENDIF] oder als Maske <CONDITION> THEN <VAR_NAME> [ELSE <VAR_NAME>]	*Bedingter Kommandoblock.*
LABEL		/NOUSER	xpos, ypos, center, angle, size, text	*-> PPL %LABEL*
LET				*-> DEFINE VARIABLE*
LIST		/I/J/K/L/M/N /X/Y/Z/T/E/F /D /ILIMITS /JLIMITS /KLIMITS /LLIMITS /XLIMITS /YLIMITS /ZLIMITS /TLIMITS /APPEND /FILE /FORMAT /HEADING /NOHEAD /TITLE /ORDER /RIGID /PRECISION /CLOBBER /SINGLE /QUIET /WIDTH /EDGES /BOUNDS /NOROWLAB /KEEP_AXISNAMES /OUTTYPE /NCFORMAT /XCHUNK/YCHUNK/ZCHUNK/TCHUNK/DEFLATE/SHUFFLE /ENDIAN	<EXPRESSION> [,<EXPRESSION> [,...]]	*Wertet bestimmte Ausdrücke oder Variablen aus aud listet sie auf der Konsole oder in eine Datei.*
LOAD		/I/J/K/L/M/N /X/Y/Z/T/E/F /D /NAME /PERMANENT /TEMPORARY	<EXPRESSION> [,<EXPRESSION> [,...]]	*Lädt Datensätze in den Speicher vor.*
MESSAGE		/CONTINUE /QUIET /JOURNAL /ERROR	<TEXT>	*Ausgabe in einem Skript*
PALETTE				*-> PPL SHASET SPECTRUM=*
PATTERN				*-> PPL PATSET PATTERN=*
PAUSE				*-> MESSAGE*
PLOT		/I/J/K/L/M/N /X/Y/Z/T/E/F /OVERLAY/ SET_UP /FRAME /D/ TRANPOSE/ VS/ SYMBOL/ NOLABEL /LINE /COLOR /THICKNES /SIZE /HLIMITS /VLIMITS /TITLE /STEP /NOAXES /DASH /NOYADJUS /AXES /HLOG /VLOG /NOKEY /KEY /RIBBON /LEVELS /PALETTE /FAST /MISSING /OPACITY	<EXPRESSION> [,<EXPRESSION> [,...]] oder Scatter-Plot: <XVAR>,<YVAR> oder Ribbon-Plot: <EXPRESSION>,<EXPRESSION-FOR_COLOR> oder Scatter-Ribbon-Plot: <XVAR>,<YVAR><EXPRESSION_FOR_COLOR>	*Eingacer Linien- oder Scatter-Plot. Plotlinie kann auch eingefärbt werdne (Ribbon).*
POLYGON		/I/J/K/L/M/N /X/Y/Z/T/E/F /OVERLAY /SET_UP /FRAME/D /TRANSPOSE /COORD_AX /NOLABELS /LEVELS /LINE /COLOR /PALETTE /TITLE /THICKNESS /NOAXES /PATTERN /FILL /KEY /NOKEY /HLIMITS /VLIMITS /MODULO /OPACITY	<X-VERTICES>,<Y-VERTICES> [,<EXPRESSION-FOR-COLOR>]	*Plottet einen Polygonzug. Kann als Fläche mit Farbe oder mit einer Schrafur gefüllt werden.*
PPLUS		/RESET	<PPLUS_COMMAND>	*Zusatzkommandos zum Modifizieren der Grafik-Eigenschaften, wie Fareben, Beschriftung oder ähnliches.*
QUERY				*Überprüft notwendige Argumente in einem Skript.*
QUIT				*-> EXIT*
REPEAT		/I/J/K/L/M/N /X/Y/Z/T/E/F /RANGE= /NAME= /ANIMATE/LOOP=	<COMMAND> oder Mehrzeiliges Kommand: (<COMMAND_1>; <COMMAND_2>;\ <COMMAND_3>;)	*Führt eine Schleife entlang eines Index aus.*
SAVE				*-> LIST/FORMAT=CDF*
SAY				*-> MESSAGE/CONTINUE*
SET	**ATTRIBUTE**		<VARNAME>.<ATTNAME>	*Setzt Wert für NetCDF Attribut fest.*
	AXIS		<AXIS_NAME>	*Wählt bestimmte Ahcse als Default aus.*
	DATA_SET		<DATASET> [,<DATASET> ,[...]]	*Wählt einen Datensatz als Default aus.*
	EXPRESSION		<EXPRESSION> [,<EXPRESSION> [,...]]	*Legt einen Default-Ausdruck fest.*
	GIFFILE		<FILE_NAME>	*Legt einen gif-Dateinamen fest.*
	GRID		<GRID_OR_VARIABLE_NAME>	
	LIST			
	MEMORY			
	MODE		<MODE> siehe Mode-Tabelle	
	MOVIE	/COMPRESS /FILE /LASER /START		

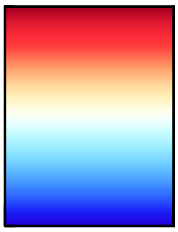
	NCCACHE	/SIZE/NELEMS/PREEMPT		
	REDIRECT	/TEE /JOURNAL /FILE= /APPEND /CLOBBER		
	REGION	/I/J/K/L/M/N /X/Y/Z/T/E/F /DI/DJ/DK/DL/DM/DN /DX/DY/DZ/DT/DE/DF	[<REGION_NAME>]	
	VARIABLE	/BAD /GRID /TITLE /UNIT /DATASET/NAME/OFFSET/SCALE/OUTTYPE	[<VARIABLE_NAME>]	
	VIEWPORT		<VIEW_NAME>	
	WINDOW	/ASPECT /CLEAR /LOCATION /NEW /SIZE /TITLE /QUALITY /ANTIALIAS | /NOANTIALIAS /COLOR /OPACITY=n /THICKEN=f /XINCHES=f/YINCHES=f /XPIXELS=n/YPIXELS=n /TEXTPROM=f	<WINDOW_NUMBER>	
SHADE		/I/J/K/L/M/N /X/Y/Z/T/E/F /D /FRAME /KEY /LEVELS /LINE /NOAXIS /NOKEY /NOLABELS /OVERLAY /PALETTE /PATTERN /SET_UP /TITLE /TRANSPOSE /MODULO /HLIMITS /VLIMITS /AXES /TRIM /OPACITY	<EXPRESSION> oder für Krummlinige Felder: <EXPRESSION>,<XCOORDS>,<YCOORDS>	*Ziechnet Werte eines 2D-Feldes als Pixel.*
SHOW		/ALL		
	ALIAS		[<ALIAS_NAME>]	
	ATTRIBUTE	/ALL /DATASET /OUTPUT	<VARNAME>.<ATTNAME> oder <VARNAME> oder <DATASET>	
	AXIS	/I=/J=/K=/L=/M=/N=/X=/Y=/Z=/T=/E=/F=/XML/OUTFILE/APPEND/CLOBBER	<AXIS_NAME>	
	COMMANDS	/ALL	<COMMAND_NAME_OR_PARTIAL_COMMAND>	
	DATA_SET	/ALL /BRIEF /FILES /FULL /VARIABLE /ATTRIBUTES /XML /OUTFILE /APPEND /CLOBBER	<DATASET> [,<DATASET> ,[...]]	
	EXPRESSION			
	FUNCTION	/ALL /BRIEF /EXTERNAL /INTERNAL /DETAILS	[<FUNCTION_NAME>]	
	GRID	/I/J/K/L/M/N /X/Y/Z/T/E/F /ALL /DYNAMIC	<VARIABLE_OR_GRIDNAME> [,<VARIABLE_OR_GRIDNAME> [,...]]	
	LIST			
	MEMORY	/ALL/FREE/PERMANENT/TEMPORARY		
	MODE		<MODE_NAME> [,<MODE_NAME> [,...]]	
	MOVIE			
	NCCACHE			
	QUERIES			
	REGION		[<REGION_NAME>]	
	SYMBOL	/ALL	[<SYMBOL_NAME>]	
	TRANSFORM			
	VARIABLES	/ALL /DATASET /DIAGNOSTIC /USER /XML /OUTFILE /APPEND /CLOBBER /TREE	[<VAR_NAME>]	
	VIEWPORT		<VIEW_NAME> [,<VIEW_NAME> [,...]]	
	WINDOWS			
SPAWN			<UNIX_SHELL_COMMAND>	*Führt ein Unix-Kommando aus, ohne dass ferret verlassen werden muss.*
STATISTICS		/I/J/K/L/M/N X/Y/Z/T/E/F /D /BRIEF	<EXPRESSION> [,<EXPRESSION> [,...]]	*Berechnet einfache statistische Werte eines Ausdrucks und gibt sie aus.*
UNALIAS				*-> CANCEL ALIAS*
VECTOR		/I/J/K/L/M/N /X/Y/Z/T/E/F /D /ASPECT /FRAME /LENGTH /NOAXIS /NOLABELS /OVERLAY /PEN /SET_UP /TITLE /COLOR /TRANSPOSE /HLIMITS /VLIMITS /XSKIP /YSKIP /MODULO /OPACITY	<U_EXPRESSION>,<V_EXPRESSION> oder <U_EXPRESSION>,<V_EXPRESSION>,<XCOORDS>,<YCOORDS>	*Zeichnet Vektor-Pfeile.*

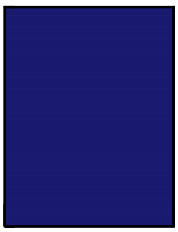
B Farb-Paletten

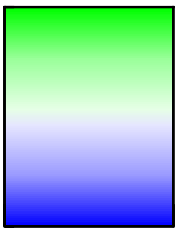

black

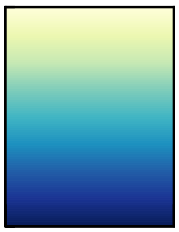

blue_brown

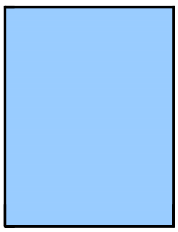

blue_darkorange

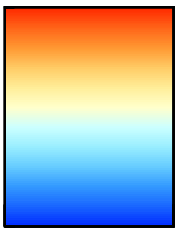

blue_darkred

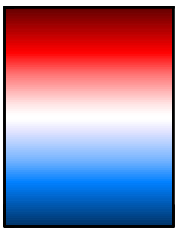

blue_dark

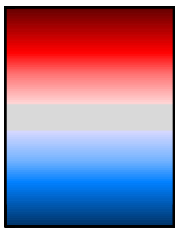

blue_green

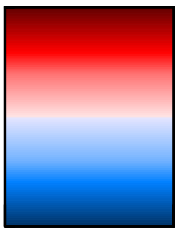

blue_green_yellow

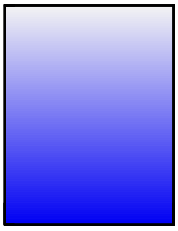

blue_light

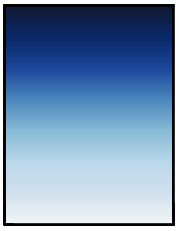

blue_orange

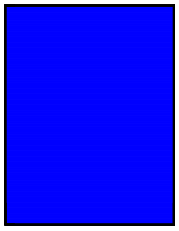

blue_red_centered

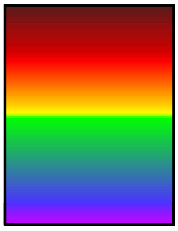

blue_red_gray_centered

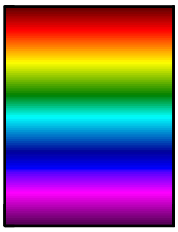

blue_red_nowhite_centered

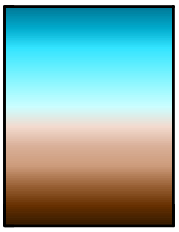

bluescale

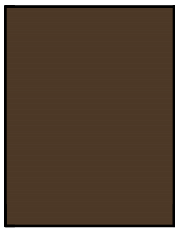

blues_cmyk

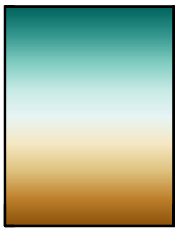

blue

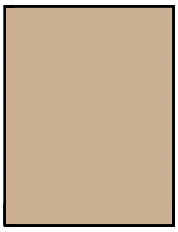

bright_centered

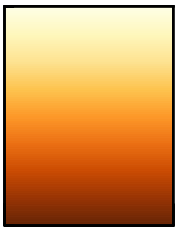

broad

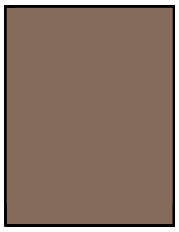

brown_blue

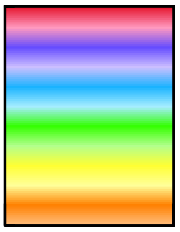

brown_dark

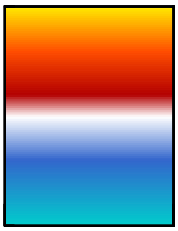

brown_green

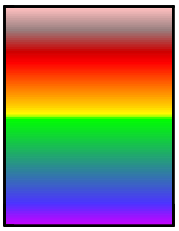

brown_light

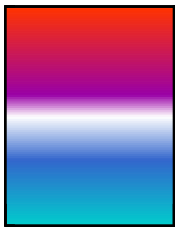

brown_orange_yellow

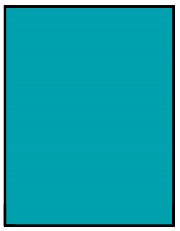

brown

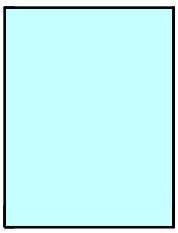

categorical_12_step

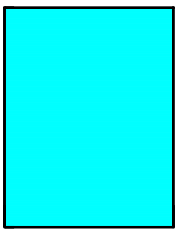

centered_diff

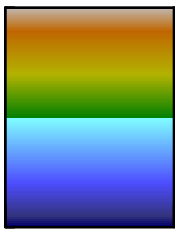

centered

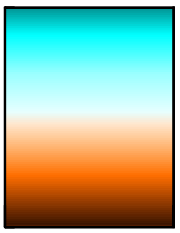

centered_turq_red_purple

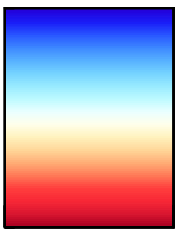

cyan_dark

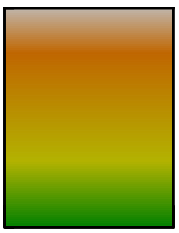

cyan_light

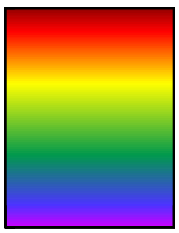

cyan

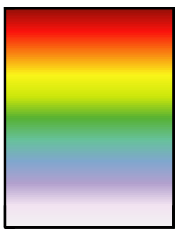

dark_land_sea

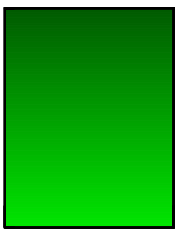

darkorange_blue

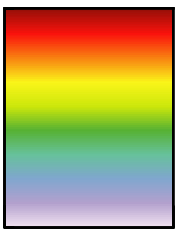

darkred_blue

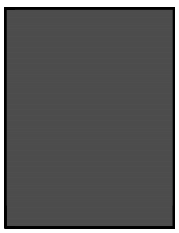

dark_terrestrial

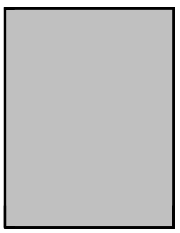

default

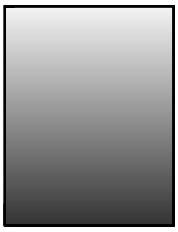

dynamic_cmyk

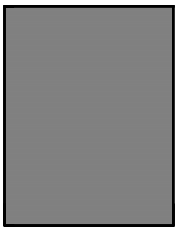

etop_values

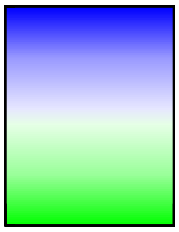

exciting_cmyk

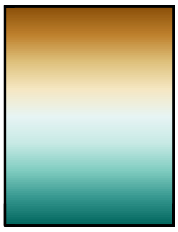

gray_dark

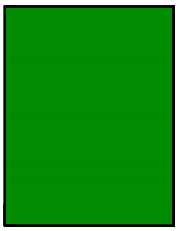

gray_light

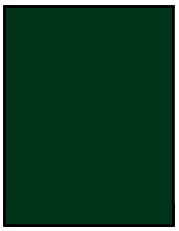

grayscale

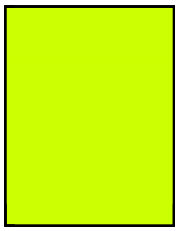

gray

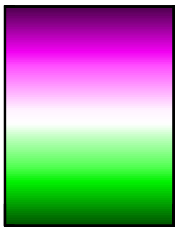

green_blue

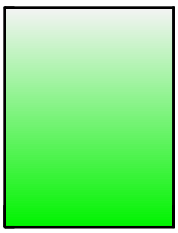

green_brown


green_dark

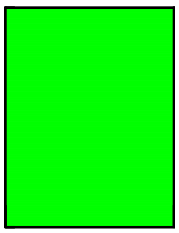

green_deep

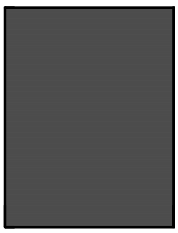

green_light

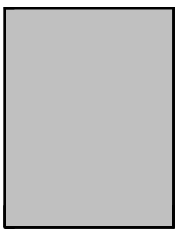

green_magenta

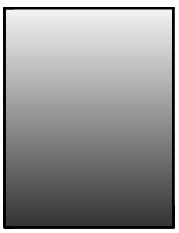

greenscale

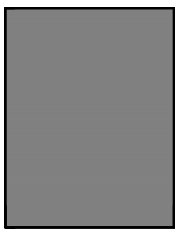

greens_cmyk

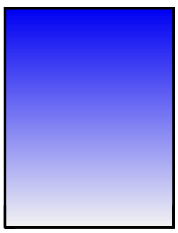

green

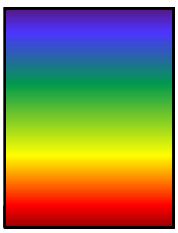

grey_dark

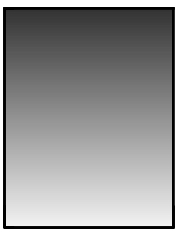

grey_light

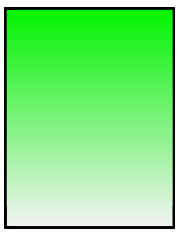

greyscale

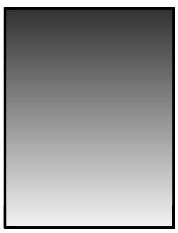

grey

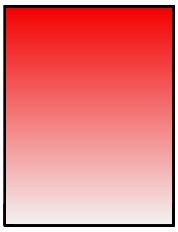

inverse_bluescale

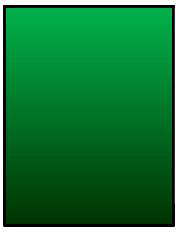

inverse_dark_rainbow

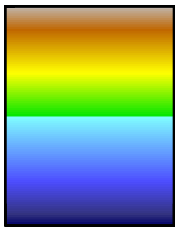

inverse_grayscale

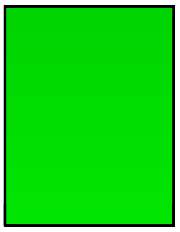

inverse_greenscale

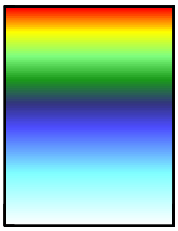

inverse_greyscale

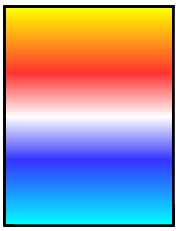

inverse_redscale

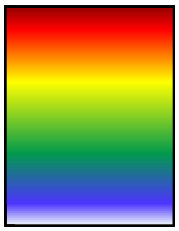

land_sea3

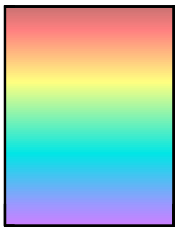

land_sea

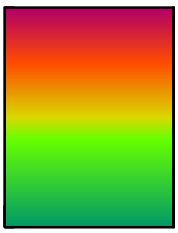

land_sea_values

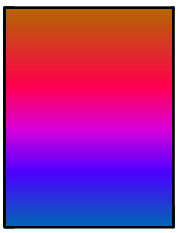

light_bottom

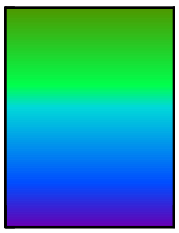

light_centered

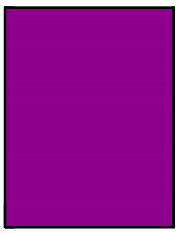

lightgray_bottom

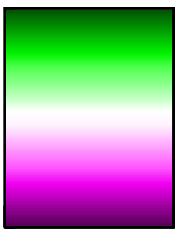

light_rainbow

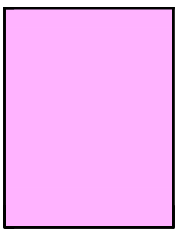

low_blue

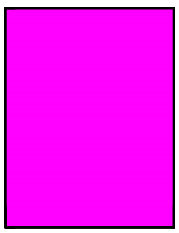

low_green

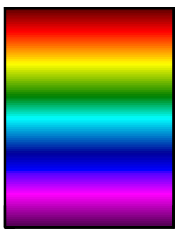

low_red


magenta_dark

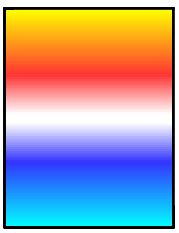

magenta_green

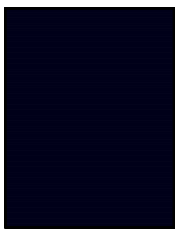

magenta_light

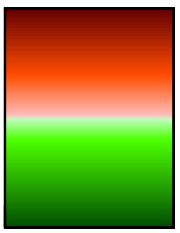

magenta

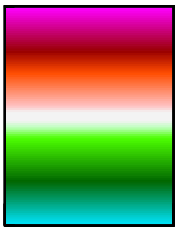

modulo

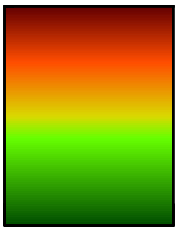

more_by_levels

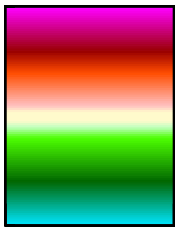

narrow_light_centered

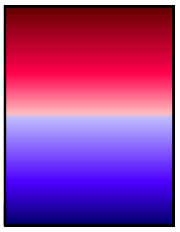

navy

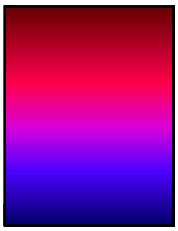

no_blue_centered

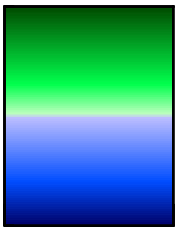

no_blue_grey_centered

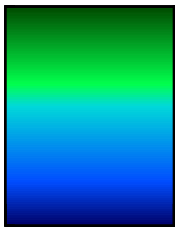

no_blue

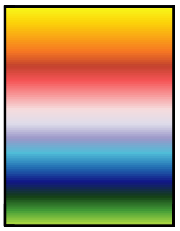

no_blue_yellow_centered

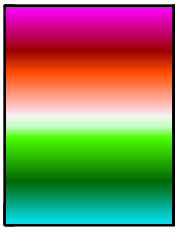

no_green_centered

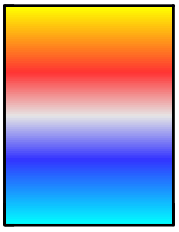

no_green

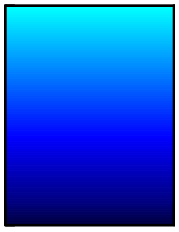

no_red_centered

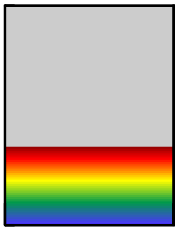

no_red

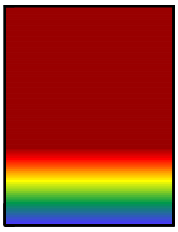

norm_cent_cmyk

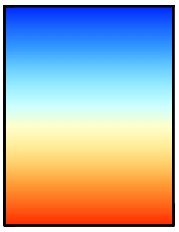

no_white_greenpink_centered

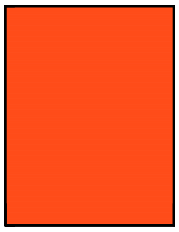

no_white_light_centered

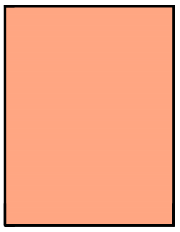

ocean_blue

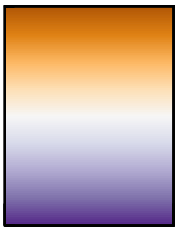

ocean_temp_bounds

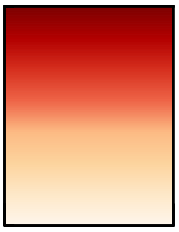

ocean_temp

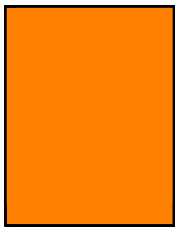

orange_blue

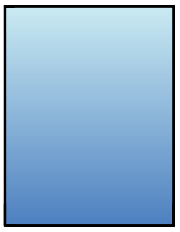

orange_dark

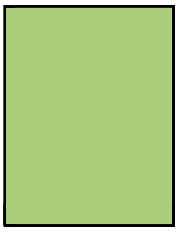

orange_light

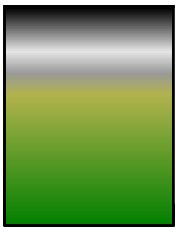

orange_purple

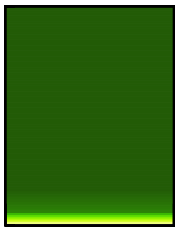

orange_red

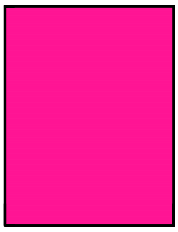

orange

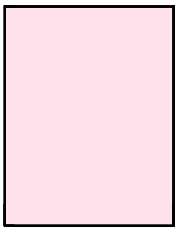

osmc_bluescale

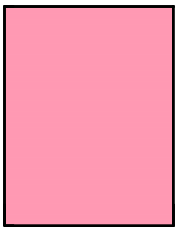

osmc_land

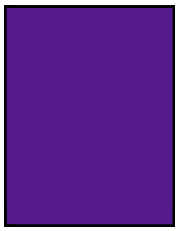

osmc_terrestrial

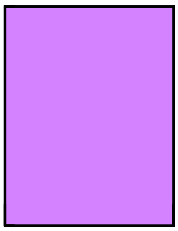

paleo_psdi_value

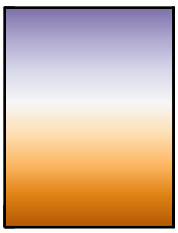

pink_dark


pink_light

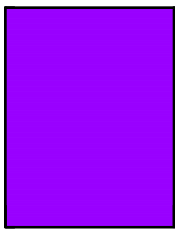

pink

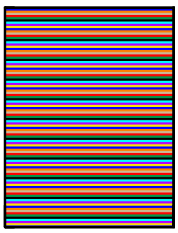

purple_dark

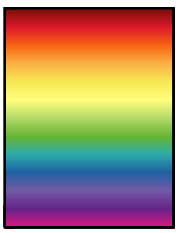

purple_light

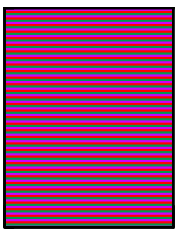

purple_orange

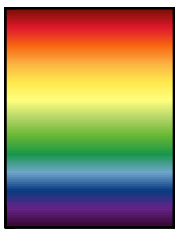

purple_red

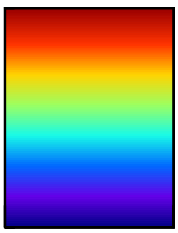

purple

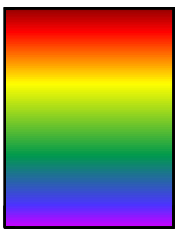

QC_by_level

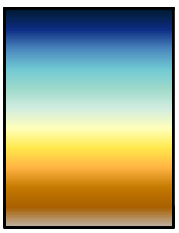

rainbow2_cmyk

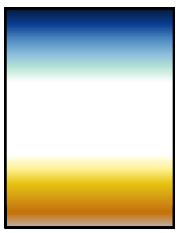

rainbow_by_levels

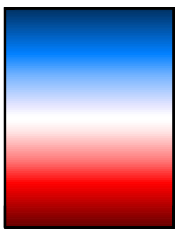

rainbow_cmyk

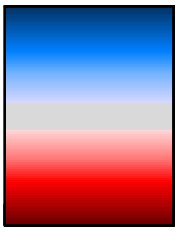

rainbow_rgb

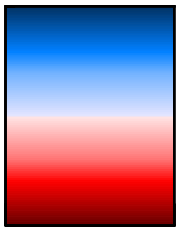

rainbow

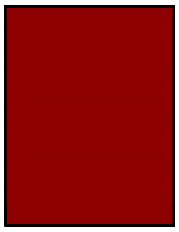

rain_cmyk

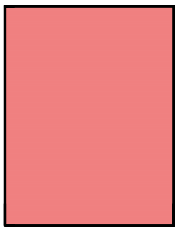

rain_hole_cmyk

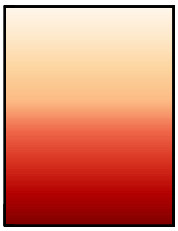

red_blue_centered

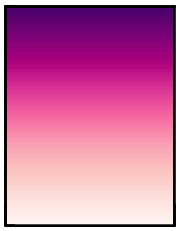

red_blue_gray_centered

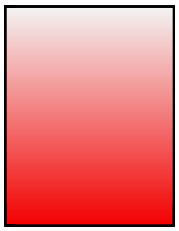

red_blue_nowhite_centered

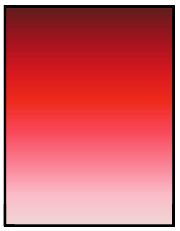

red_dark

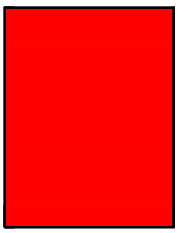

red_light

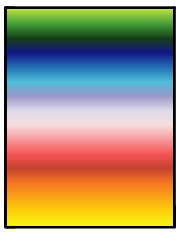

red_orange

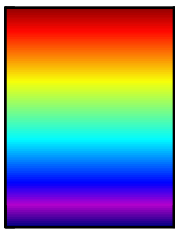

red_purple

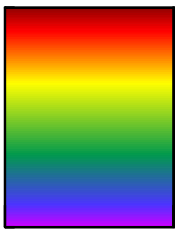

redscale

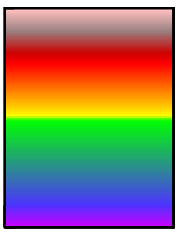

reds_cmyk

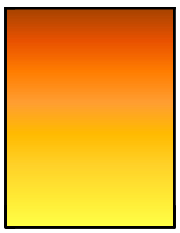

red

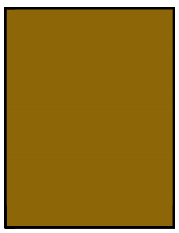

rev_cent_cmyk

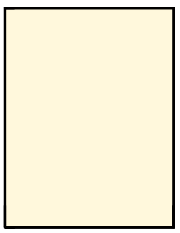

rnb2

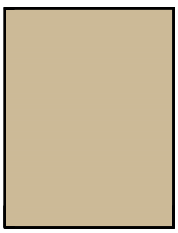

rnb

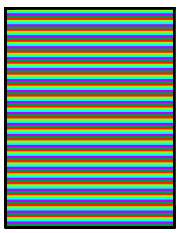

saz2

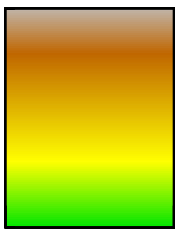

sunny_cmyk

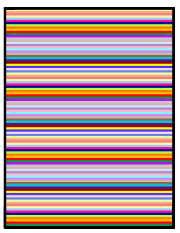

tan_dark

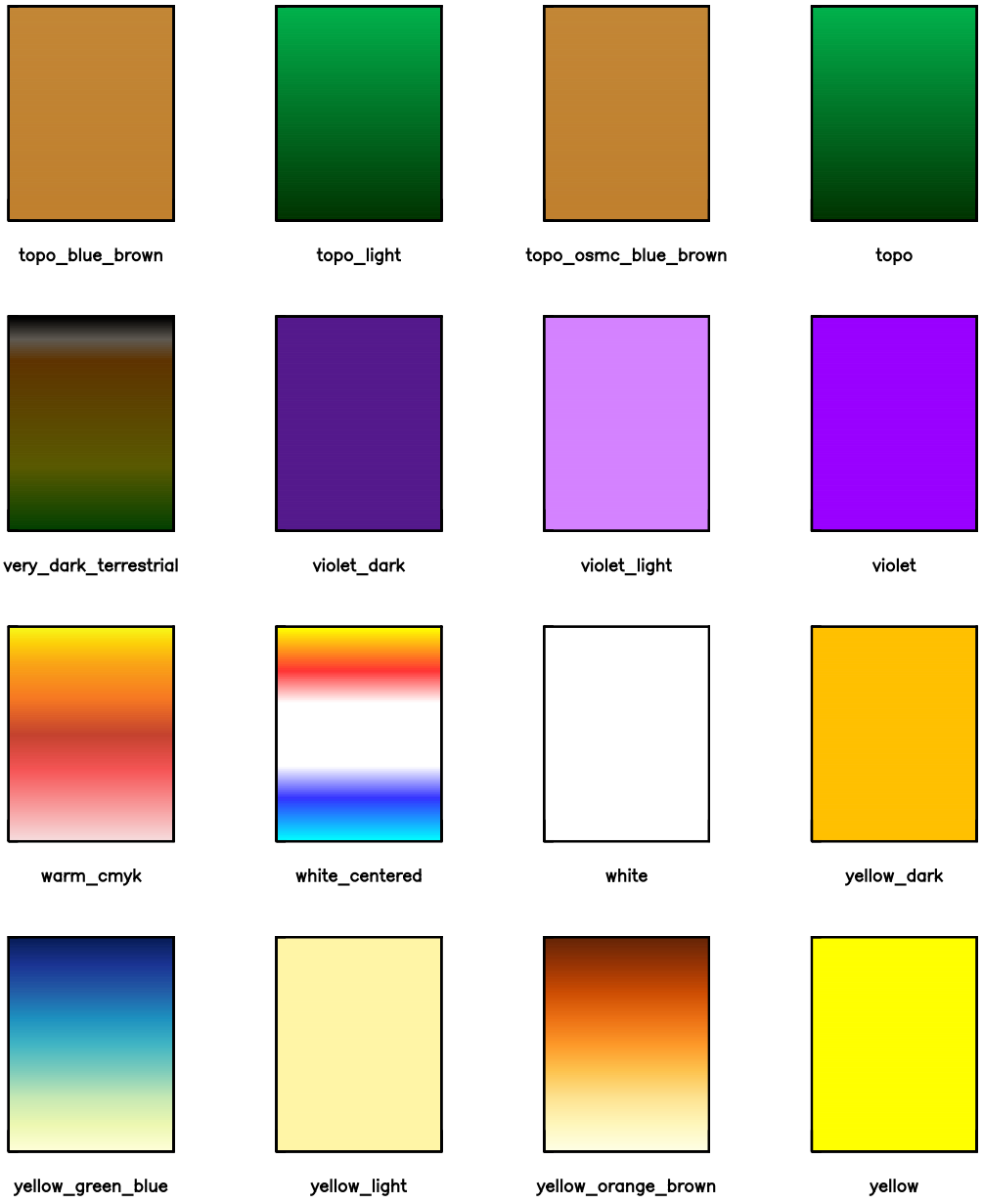

tan_light


tan


ten_by_levels


terrestrial


thirty_by_levels



Diese Palettenübersicht wurde mit fogenden Kommandos erstellt (unter Verwendung des Skriptes showpal.jnl auf Seite 84):

```
# In der UNIX-Shell:
Fpalette '*' | grep .spk | sed 's/\.spk:///'> ./fer_all_pal.lst

pyferret -nojnl
go showpal
```

showpal.jnl

```
let/title="scale" a=(y[j=1:100]-1)*(x[i=1:2]*0+1)
se wi/asp=1.4/qual=high
go portraitNxN 4 5 30 30
ca mo logo
col/var=PAL "./fer_all_pal.lst"
ca vi
ppl shaset reset
repeat/name=q/range=1:`PAL,RETURN=IEND`:20 (\
  repeat/name=v/range=1:20:1 (\
    say `q`,`v+q-1`;se vi `v`; \
    let vqm1=`v+q-1`;\
    let PALNUM=`PAL,RETURN=IEND`;\
    IF `vqm1 le PALNUM` THEN ;\
      def sym PAL=PAL[i=`v+q-1`];\
      shade/lev=.5d/nokey/pal=`($PAL)`/set/title="" a;\
      ppl axlabp,0,0 ;\
      ppl tics,0,0,0,0 ;\
      ppl shade;\
      label 1.5 -18 0 0 .14 @AS`($PAL)` ;\
    ENDIF ;\
  );\
  frame/file="./fer_pal_`q`.pdf";\
  ca vi;\
  ppl shaset reset;\
)
```

C Klimatologische Zeitachsen in climatological_axes.cdf

```
yes? use climatological_axes
 *** NOTE: regarding /usr/local/PyFerret_1.0.2/go/climatological_axes.cdf ...
 *** NOTE: Climatological axes SEASONAL_REG, MONTH_REG, MONTH_IRREG, MONTH_GREGORIAN, MONTH_NOLEAP
     , MONTH_360_DAY, MONTH_ALL_LEAP  defined
yes? sh da
     currently SET data sets:
    1> /usr/local/PyFerret_1.0.2/go/climatological_axes.cdf  (default)
 name     title                               I         J         K         L


yes? sh ax/t SEASONAL_REG, MONTH_REG, MONTH_IRREG, MONTH_GREGORIAN, MONTH_NOLEAP, \
            MONTH_360_DAY, MONTH_ALL_LEAP
...
```

SEASONAL_REG

4 Zeitpunkte: JAN-FEB-MAR, APR-MAI-JUN, JUL-AUG-SEP, OCT-NOV-DEC; reguläre (auf Basis gleich-langer Monate) modulo-Achse in Stunden seit 01-JAN-0000; Gregorianischer Kalender.

```
 name       axis              # pts   start                end
 SEASONAL_REG TIME              4mr   15-FEB 15:43         15-NOV 14:05
T0 = 01-JAN-0000 00:00:00
   Axis span (to cell edges) = 8765.82 (modulo length = axis span)

       L     T                  TBOX      TBOXLO                TSTEP (HOURS)
       1>  15-FEB 15:43:39      2191.455   01-JAN 00:00:00        1095.727
       2>  16-MAY 23:10:57      2191.455   01-APR 07:27:18        3287.182
       3>  16-AUG 06:38:15      2191.455   01-JUL 14:54:36        5478.637
       4>  15-NOV 14:05:33      2191.455   30-SEP 22:21:54        7670.092
```

MONTH_REG

12 Zeitpunkte: in der Mitte eines jeden Monats; reguläre (auf Basis gleich-langer Monate) modulo-Achse in Stunden seit 01-JAN-0000; Gregorianischer Kalender.

```
name       axis              # pts   start                end
MONTH_REG TIME                12mr   16-JAN 06:00         16-DEC 01:20
T0 = 01-JAN-0000 00:00:00
   Axis span (to cell edges) = 8765.82 (modulo length = axis span)

       L     T                  TBOX      TBOXLO                TSTEP (hour)
       1>  16-JAN 06:00:00      730.485    01-JAN 00:45:27          366
       2>  15-FEB 16:29:06      730.485    31-JAN 11:14:33         1096.485
       3>  17-MAR 02:58:12      730.485    01-MAR 21:43:39         1826.97
       4>  16-APR 13:27:18      730.485    01-APR 08:12:45         2557.455
       5>  16-MAY 23:56:24      730.485    01-MAY 18:41:51         3287.94
       6>  16-JUN 10:25:30      730.485    01-JUN 05:10:57         4018.425
       7>  16-JUL 20:54:36      730.485    01-JUL 15:40:02         4748.91
       8>  16-AUG 07:23:42      730.485    01-AUG 02:09:09         5479.395
       9>  15-SEP 17:52:48      730.485    31-AUG 12:38:15         6209.88
      10>  16-OCT 04:21:54      730.485    30-SEP 23:07:21         6940.365
      11>  15-NOV 14:51:00      730.485    31-OCT 09:36:27         7670.85
      12>  16-DEC 01:20:05      730.485    30-NOV 20:05:32         8401.335
```

MONTH_IRREG, MONTH_GREGORIAN

12 Zeitpunkte: in der Mitte eines jeden Monats; irreguläre (basierend auf variablen Monatslängen) modulo-Achse in Tagen seit 01-JAN-0000; Gregorianischer Kalender.

```
name       axis              # pts   start                end
 MONTH_IRREG TIME              12mi   16-JAN 12:00         15-DEC 17:49
T0 = 01-JAN-0000 00:00:00
   Axis span (to cell edges) = 365.2425 (modulo length = axis span)

       L     T                   TBOX      TBOXLO                TSTEP (DAYS)
       1>  16-JAN 12:00:00       31         01-JAN 00:00:00         15.5
       2>  15-FEB 02:54:36       28.2425    01-FEB 00:00:00         45.12125
       3>  15-MAR 17:49:12       31         29-FEB 05:49:12         74.7425
       4>  15-APR 05:49:12       30         31-MAR 05:49:12         105.2425
       5>  15-MAY 17:49:12       31         30-APR 05:49:12         135.7425
       6>  15-JUN 05:49:12       30         31-MAY 05:49:12         166.2425
       7>  15-JUL 17:49:12       31         30-JUN 05:49:12         196.7425
       8>  15-AUG 17:49:12       31         31-JUL 05:49:12         227.7425
       9>  15-SEP 05:49:12       30         31-AUG 05:49:12         258.2425
      10>  15-OCT 17:49:12       31         30-SEP 05:49:12         288.7425
      11>  15-NOV 05:49:12       30         31-OCT 05:49:12         319.2425
      12>  15-DEC 17:49:12       31         30-NOV 05:49:12         349.7425
```

MONTH_NOLEAP

12 Zeitpunkte: in der Mitte eines jeden Monats; irreguläre (basierend auf variablen Monatslängen) modulo-Achse in Tagen seit 01-JAN-0000; 365-Tage Kalender; keine Schaltjahre (NOLEAP).

```
name       axis              # pts   start                end
 MONTH_NOLEAP TIME             12mi   16-JAN 12:00         16-DEC 12:00
T0 = 01-JAN-0000 00:00:00
CALENDAR = NOLEAP
   Axis span (to cell edges) = 365 (modulo length = axis span)

       L     T                   TBOX      TBOXLO                TSTEP (days)
       1>  16-JAN 12:00:00       31         01-JAN 00:00:00         15.5
       2>  15-FEB 00:00:00       28         01-FEB 00:00:00         45
       3>  16-MAR 12:00:00       31         01-MAR 00:00:00         74.5
       4>  16-APR 00:00:00       30         01-APR 00:00:00         105
       5>  16-MAY 12:00:00       31         01-MAY 00:00:00         135.5
       6>  16-JUN 00:00:00       30         01-JUN 00:00:00         166
       7>  16-JUL 12:00:00       31         01-JUL 00:00:00         196.5
       8>  16-AUG 12:00:00       31         01-AUG 00:00:00         227.5
       9>  16-SEP 00:00:00       30         01-SEP 00:00:00         258
      10>  16-OCT 12:00:00       31         01-OCT 00:00:00         288.5
      11>  16-NOV 00:00:00       30         01-NOV 00:00:00         319
      12>  16-DEC 12:00:00       31         01-DEC 00:00:00         349.5
```

MONTH_360_DAY

12 Zeitpunkte: in der Mitte eines jeden Monats; reguläre (basierend auf gleichlangen Monaten) modulo-Achse in Tagen seit 01-JAN-0000; 360-Tage Kalender.

```
name       axis              # pts   start                end
 MONTH_360_DAY TIME            12mr   16-JAN 00:00         16-DEC 00:00
T0 = 01-JAN-0000 00:00:00
CALENDAR = 360_DAY
   Axis span (to cell edges) = 360 (modulo length = axis span)

       L     T                   TBOX      TBOXLO                TSTEP (days)
       1>  16-JAN 00:00:00       30         01-JAN 00:00:00         15
       2>  16-FEB 00:00:00       30         01-FEB 00:00:00         45
       3>  16-MAR 00:00:00       30         01-MAR 00:00:00         75
       4>  16-APR 00:00:00       30         01-APR 00:00:00         105
       5>  16-MAY 00:00:00       30         01-MAY 00:00:00         135
       6>  16-JUN 00:00:00       30         01-JUN 00:00:00         165
       7>  16-JUL 00:00:00       30         01-JUL 00:00:00         195
       8>  16-AUG 00:00:00       30         01-AUG 00:00:00         225
       9>  16-SEP 00:00:00       30         01-SEP 00:00:00         255
      10>  16-OCT 00:00:00       30         01-OCT 00:00:00         285
      11>  16-NOV 00:00:00       30         01-NOV 00:00:00         315
      12>  16-DEC 00:00:00       30         01-DEC 00:00:00         345
```

MONTH_ALL_LEAP

```
name       axis              # pts   start                end
 MONTH_ALL_LEAP TIME           12mi   16-JAN 12:00         16-DEC 12:00
T0 = 01-JAN-0000 00:00:00
CALENDAR = ALL_LEAP
   Axis span (to cell edges) = 366 (modulo length = axis span)

       L     T                   TBOX      TBOXLO                TSTEP (days)
       1>  16-JAN 12:00:00       31         01-JAN 00:00:00         15.5
       2>  15-FEB 12:00:00       29         01-FEB 00:00:00         45.5
       3>  16-MAR 12:00:00       31         01-MAR 00:00:00         75.5
       4>  16-APR 00:00:00       30         01-APR 00:00:00         106
       5>  16-MAY 12:00:00       31         01-MAY 00:00:00         136.5
       6>  16-JUN 00:00:00       30         01-JUN 00:00:00         167
       7>  16-JUL 12:00:00       31         01-JUL 00:00:00         197.5
       8>  16-AUG 12:00:00       31         01-AUG 00:00:00         228.5
       9>  16-SEP 00:00:00       30         01-SEP 00:00:00         259
      10>  16-OCT 12:00:00       31         01-OCT 00:00:00         289.5
      11>  16-NOV 00:00:00       30         01-NOV 00:00:00         320
      12>  16-DEC 12:00:00       31         01-DEC 00:00:00         350.5
```

D Go Scripte

Skript Name	Erläuterung
argo_zt.jnl	
bar_chart_demo.jnl	demonstrate usage of bar charts
bar_chart1.jnl	Make a bar chart using color fill (not suitable for Z axis)
bar_chart2.jnl	Make a bar chart using hollow rectangles
bar_chart3.jnl	Produce a color-filled bar chart
basemap.jnl	Plot a base map of the earth (solid, outline, or detailed)
bering_coast.jnl	line-style overlay of bering sea & asian coastlines
bering_iso100.jnl	overlay of bering sea & asian 100m isobath
bering_iso1000.jnl	overlay of bering sea & asian 1000m isobath
binary_read_demo.jnl	
black.jnl	set video background to black, foreground to white
bold.jnl	set up PLOT+ and FERRET to produce bolder-looking plots
box_plot.jnl	produce a plot with "bare" axes (no tics, no labels)
box.jnl	Overlay a colored box outline given the corner coordinates
bullseye_box.jnl	draw box around bullseye search region
bullseye.jnl	locate and show a bullseye in a region of data
call_stack_stick.jnl	
call_stack.jnl	
ccbar_demo.jnl	demonstrate continuous colorbar on plots that use
ccbar.jnl	Continuous colorbar script, uses a shade plot in a viewport to make a colorbar
centered_vectors.jnl	Vector plot with coords at vector midpoints
cleanup_text.jnl	cleanup FERRET and PLOT+ settings following GO setup_text
coads_demo.jnl	introduces and briefly explores the COADS climatological data
color_vector_demo.jnl	
color_vector.jnl	
compass_rose.jnl	
constant_array_demo.jnl	Demonstration of the {1, 3, 5} "constant array" syntax
convert_to_fnoc_2d.jnl	extract (sample) lat/lon data onto the FNOC polar grid
convert_to_polar_2d.jnl	extract (sample) data for a 2D polar plot
create_KMZ_groundoverlay_1.jnl	
create_KMZ_groundoverlay_2.jnl	
create_KMZ_groundoverlay_3.jnl	
create_KMZ_groundoverlay_4.jnl	
create_KMZ_tour_1.jnl	
create_KMZ_tour_2.jnl	
create_KMZ_tour_3.jnl	
create_KMZ_tour_4.jnl	
create_KMZ.jnl	
custom_contour_demo.jnl	show customizable features of PLOT+ contouring within FERRET
datestring.jnl	
def_monthaxis_days.jnl	Define a time axis in days in specific calendar
depth_to_density_demo.jnl	
depth_to_density_weq_demo.jnl	show contour with a user-defined variable as an axis
digitize.jnl	Click cursor on plot – get value or surrounding block
dods_demo.jnl	Demo: how to use Ferret and DODS to access remote Datasets
draw_it.jnl	
dynamic_height.jnl	Define FERRET variables for dynamic height calculations
edit_data_file_demo.jnl	demo of a technique for "hand-editing" a variable
ef_eof_demo.jnl	Demonstration of computing EOFs using the
ef_eofsvd_demo.jnl	Demonstration of computing EOFs using the
ef_fft_demo.jnl	Demonstration of external functions for FFT
ef_sort_demo.jnl	examples of the sorting external functions for FERRET.

Skript Name	Erläuterung
ef_wv5d_demo.jnl	! Description: examples of the external function to write Vis5D files
ellipse.jnl	Overlay an ellipse outline from inscribing rectangle coordinates
error_bars_demo.jnl	demonstrate usage of error bars
error_bars.jnl	Overlay error bars on a plot
exact_colors.jnl	Setup FERRET and PLOT+ to modify single SHADE or CONT colors
extremum.jnl	Annotate an extremum for a region on a plot
fft2drun.jnl	computes and plots running power spectrum for variable "specvar"
file_reading_demo.jnl	shows examples of reading data from ASCII files
fill_between.jnl	Fill between two curves or a curve and a constant value
find_bullseye.jnl	find bullseye when told if max or min (subroutine)
fland.jnl	Plot or overlay an area-filled map of the continents
flip_palette.jnl	
fnoc_2d.jnl	produce a 2D polar stereographic plot
fnoc_demo.jnl	explores the Navy's FNOC surface marine wind field.
fnoc_map_grid.jnl	define the map grid for the FNOC polar 63x63 grid
fnoc_verify.jnl	
focean.jnl	Plot or overlay an area-filled map of the oceans
frequency_histogram.jnl	produce a 1D frequency histogram
frequency_histogram2.jnl	produce a 1D frequency histogram, using Ferret functions to
gmt_power.jnl	define a variable for the power spectrum of a time series
graticule.jnl	Set the plot axis style to use a graticule (see GO tics)
graticules_demo.jnl	
gratxt.jnl	create an overlay of XT graticule lines to show the axis values
gratxy.jnl	create an overlay of XY graticule lines to show the axis values
gratxz.jnl	create an overlay of XZ graticule lines to show the axis values
gratyt.jnl	create an overlay of YT graticule lines to show the axis values
gratyz.jnl	create an overlay of YZ graticule lines to show the axis values
gratzt.jnl	create an overlay of TZ graticule lines to show the axis values
great_circle.jnl	Overlay a great circle arc on a lat/lon plot
gridlines.jnl	plot over the grid of a plot
gridxt.jnl	create an overlay of IL graticule lines to show the grid points
gridxy.jnl	create an overlay of IJ graticule lines to show the grid points
gridxz.jnl	create an overlay of IK graticule lines to show the grid points
gridyt.jnl	create an overlay of JL graticule lines to show the grid points
gridyz.jnl	create an overlay of JK graticule lines to show the grid points
gridzt.jnl	create an overlay of KL graticule lines to show the grid points
histogram_pdf.jnl	generate and plot a frequency histogram from a FERRET variable
histogram.jnl	generate and plot a frequency histogram from a FERRET variable
image_to_kml.jnl	
janfebmar_time_axis.jnl	a time axis of Jan,Feb,Mar only from successive years
label_hi_lo.jnl	overlay labels for local extrema with H and L or the
labelvalues.jnl	Label locations on a 2-D plot with values and optional text
land_detail_demo.jnl	
land_detail.jnl	Plot outlines of continents, countries, states, and/or rivers.
land_full.jnl	Plot outlines of continents, countries, states, and/or rivers.
land.jnl	Plot outlines of continents, countries, and/or states
landscape.jnl	Set up for 11 x 8.5 page
landscape1x2.jnl	Set up for 2 viewports placed horizontally on an 11 x 8.5 page
landscape2x1.jnl	Set up for 2 viewports vertically placed on an 11 x 8.5 page
landscape2x2.jnl	Set up for four viewports on an 11 x 8.5 page
landscape3x2.jnl	Set up for six viewports on an 11 x 8.5 page
landt.jnl	Overlay the GFDL/MOM/Pacific boundaries based on the TS grid

Skript Name	Erläuterung
landu.jnl	Overlay the GFDL/MOM/Pacific boundaries based on the UV grid
landz.jnl	Plot outlines of continents, countries, and/or states
langrangian_example.jnl	
left_axis_plot.jnl	Plot a single variable preparing for a 2nd axis on the right
legend.jnl	
legline_nu.jnl	
legline.jnl	
levitus_demo.jnl	explores the Climatological Atlas of the World Ocean
line_samples.jnl	draw examples of the line styles used by the PLOT command
line_segments.jnl	general routine for plotting a sequence of line segments
line_thickness.jnl	draw examples of the pen color/thickness in PLOT+
log_plot_demo.jnl	demonstrates log plots using PLOT+ in FERRET
magnify.jnl	rescale and shift the data plotting area (area inside the axes)
make_monthly_climatology.jnl	create a monthly climatology data set
margins.jnl	specify plot margins (layout of plot axes to window edges)
mask_outline.jnl	define a contour field with squared off lines for a 1-0 mask
mathematics_demo.jnl	demo of computation and display of "abstract expressions"
max_area.jnl	set the data plotting area (inside the axes) to the full window
mercator_demo.jnl	Plot a 2-panel demonstration of a Mercator projection
minmax_label_demo.jnl	
mlbase_var.jnl	
mld_dens.jnl	
mld_temp.jnl	
mp_aspect.jnl	Calculate the appropriate aspect ration
mp_bonne.jnl	Sets up variables for a Bonne map of the world
mp_craster_parabolic.jnl	Sets up variables for a Craster Parabolic map of the world
mp_demo.jnl	demo of map projection scripts, including
mp_eckert_greifendorff.jnl	Sets up variables for a Eckert-Greifendorff map of the world
mp_eckert_iii.jnl	Sets up variables for a Eckert III map of the world
mp_eckert_v.jnl	Sets up variables for a Eckert V map of the world
mp_fland.jnl	Plot or overlay an area-filled map of the continents
mp_graticule_limit.jnl	Overlay a graticule on a map, limited by a range of x and
mp_graticule.jnl	Overlay a graticule on a map.
mp_grid.jnl	Associates a data grid with a predefined map projection.
mp_hammer.jnl	Sets up variables for a Hammer map of the world
mp_label.jnl	Place a label on a map projection using user coordinates
mp_lambert_az.jnl	Sets up variables for a Lambert Azimuthal Equal Area map of the world
mp_lambert_cc.jnl	Sets up variables for a Lambert Conformal Conic map of the world
mp_lambert_cyl.jnl	Sets up variables for a Lambert Cylindrical Equal Area map of the world
mp_land_detail.jnl	Plot outlines of continents, countries, and/or states
mp_land_stripmap.jnl	This journal file is not "nice" and is provided as an example only.
mp_land.jnl	Plot outlines of continents, countries, and/or states
mp_line.jnl	Plots data using a predefined map projection.
mp_mask_outline.jnl	For a plot using map projections, define a contour field with
mp_mcbryde_fpp.jnl	Sets up variables for a McBryde-Thomas Flat-Polar Parabolic map of the world
mp_mercator.jnl	Sets up variables for a Mercator projection
mp_ocean_stripmap.jnl	This journal file is not "nice" and is provided as an example only.
mp_orthographic.jnl	Sets up variables for a Orthographic map of the world
mp_plate_caree.jnl	Sets up variables for a Plate Caree map of the world
mp_poly_vectors.jnl	
mp_polyconic.jnl	Sets up variables for a Polyconic map of the world

Skript Name	Erläuterung
mp_polymark.jnl	Plot polygons using a predefined map projection.
mp_polytube_bent.jnl	Plot a colored tube using a predefined map projection.
mp_polytube.jnl	Plot a colored tube using a predefined map projection.
mp_rotate.jnl	Sets up variables for a Rotated map of the world
mp_sinusoidal.jnl	Sets up variables for a Sinusoidal map of the world
mp_stereo_demo.jnl	demonstrate fancy map projection techniques
mp_stereographic_eq.jnl	Sets up variables for a Stereographic Equatorial map of the world
mp_stereographic_north.jnl	Sets up variables for a Stereographic North map of the world
mp_stereographic_south.jnl	Sets up variables for a Stereographic South map of the world
mp_trackplot.jnl	Plot a trackplot using a predefined map projection.
mp_vertical_perspective.jnl	Sets up variables for a Vertical Perspective map of the world
mp_viewport_aspect.jnl	Define a viewport for plotting map projections
mp_wagner_vii.jnl	Sets up variables for a Wagner VII map of the world
mp_winkel_i.jnl	Sets up variables for a Winkel I map of the world
multi_line_labels_demo.jnl	
multi_variable_demo.jnl	show some styles of plots with multiple dependent axes
multi_view_demo.jnl	
multi_view.jnl	GO file to make multiple viewports with
multi_xaxis_overlay.jnl	PLOT/Overlay a variable plot using a new X axis
multi_xaxis_plot1.jnl	Draw a plot formatted for later overlays using multiple X axes
multi_yaxis_overlay.jnl	PLOT/Overlay a variable plot using a new Y axis
multi_yaxis_plot1.jnl	Draw a plot formatted for later overlays using multiple Y axes
objective_analysis_demo.jnl	
objective.jnl	2D objective analysis of scattered data
one_climatology_month.jnl	(work routine for make_monthly_climatology)
overlay_bars.jnl	Overlay error bars on a time series (or frequency) plot
overlay_on_time_axis_demo.jnl	demonstrate PLOT/VS and POLYGON over time axes
palette_demo.jnl	demonstration of V5.0 palette capabilities
pattern_demo.jnl	
pdatekey_dms.jnl	
plot_swath_demo.jnl	demonstrate "line plots" done with swaths of color
plot_swath.jnl	Plot a color-filed swath between upper and lower bounds
plot_vectors.jnl	plot over vectors
polar_2d.jnl	produce a 2D polar stereographic plot
polar_demo.jnl	demonstrate the production of polar stereographic projections
polar_fland.jnl	Overlay solid-filled continents on a polar plot
polar_grid_fancy.jnl	overlay "fancy" radial logitude and curved latitude lines
polar_grid.jnl	overlay "quick" radial logitude and curved latitude lines
polar_land.jnl	overlay the continental outlines on a polar stereographic plot
polar_map_inv_eqns.jnl	define the equations used for polar projections
polar_south_demo.jnl	demonstrate South pole polar stereographic projections
polar_vector.jnl	produce a 2D polar stereographic vector plot
polar_vs.jnl	perform a polar PLOT/VS of 2 variables: lat long
polar.jnl	Define R and THETA from X and Y to perform (limited) polar plots
poly_arrow_key.jnl	
poly_vec_demo.jnl	demo of filled-polygon vectors
poly_vector_demo.jnl	demo of filled-polygon vectors
poly_vectors.jnl	
polymark_annotate_key.jnl	Plot symbols colored by values of a variable along a track
polymark_datekey.jnl	Plot symbols colored by values of a variable along a track
polymark_demo.jnl	demonstrate usage of polymark jnl file
polymark.jnl	Plot symbols colored by values of a variable along a track
polyshape.jnl	define xpolyshape, ypolyshape variables for polygon fills
polytube_bent_demo.jnl	

Skript Name	Erläuterung
polytube_bent.jnl	Plot a colored, sgmented tube of variable values along a plotted track.
polytube_demo.jnl	demonstrate "lagrangian" plots along a path using color fill
polytube.jnl	Plot a colored tube of variable values along a plotted track
portrait.jnl	Set up for 8.5 x 11 page size
portrait1x2.jnl	Set up for two viewports on an 8.5 x 11 page
portrait1x3.jnl	Set up for three viewports on an 8.5 x 11 page
portrait1x4.jnl	Set up for four viewports on an 8.5 x 11 page
portraitNxN.jnl	Set up an arbitrary number of evenly-spaced viewports
projected_map_grid.jnl	define the map grid for a projected plot
ptest.jnl	create a simple test line plot
quantiles.jnl	
ratio_set.jnl	Set ratio and margins
regresst.jnl	define FERRET variables for regression along the T axis
regressx.jnl	define FERRET variables for regression along the X axis
regressxy.jnl	define FERRET variables for regression along the X and Y axis
regressy.jnl	define FERRET variables for regression along the Y axis
regressz.jnl	define FERRET variables for regression along the Z axis
regridding_demo.jnl	tutorial to introduce FERRET regridding concepts
reminder.jnl	Place a reminder string on the upper left corner of a plot
remove_logo.jnl	
rgb_centered.jnl	
rgb_fireworks.jnl	a display of video color changes
rgb_grayscale.jnl	An obsolete tool equivalent to "PALETTE greyscale"
rgb_greyscale.jnl	An obsolete tool equivalent to "PALETTE greyscale"
rgb_rainbow.jnl	An obsolete tool equivalent to "PALETTE rnb"
rgb_try_em.jnl	An obsolete routine to cycle through 3 spectra
ribbon_plot_demo.jnl	Demonstration of various ribbon plotting options
right_axis_plot.jnl	Overlay a plot of one variable using an axis on the right
samplexy_demo.jnl	Use the SAMPLEXY function to create an arbitrary 2D
scalemark.jnl	Plot an "I beam" documenting a scale on a plot
scattered_vectors.jnl	Scattered vector plot from ASCII file: x,y,u,v
set_aspect.jnl	Calculate the appropriate aspect ratio for an x-y plot
set_pixel_size.jnl	set the size of the output window in pixels
setup_text.jnl	set up FERRET and PLOT+ to layout text easily
shaded_error_bar_zone.jnl	Demo of shaded (error uncertainty) region around line plot
show_88_syms.jnl	display all possible PLOT+ marks (usable in by PLOT/SYMBOL=)
show_all_patterns.jnl	
show_pattern.jnl	
show_symbols.jnl	display the PLOT+ marks used by the FERRET PLOT command
sigma_coordinate_demo_weq.jnl	demo of how to handle sigma coordinate output
sigma_coordinate_demo.jnl	demo of how to handle sigma coordinate output
single_color_palettes.jnl	
small_view.jnl	
spirograph_demo.jnl	demo of fun line plots
splash_demo.jnl	some pretty color pictures
split_z.jnl	make a CONTOUR,SHADE, or FILL plot with a split vertical axis
squares.jnl	create a filled-area test plot
squeeze_colors.jnl	Modify a palette by squeezing and stretching the color scale
stack_line.jnl	
stack_stick.jnl	
start_vis5d.jnl	Set filename and start vis5d .
statistics_demo.jnl	demo of some sample distribution functions and plots

Skript Name	Erläuterung
stick_vectors_key.jnl	Make a stick vector plot of a line of U,V values
stick_vectors.jnl	Make a stick vector plot of a line of U,V values
surface_uv_from_ssh.jnl	
symbol_demo.jnl	
taylor_agraticule.jnl	Plot graticules (lines) at specified angles (relative to correlation)
taylor_example1.jnl	
taylor_example2.jnl	
taylor_example3.jnl	
taylor_frame_accurate.jnl	
taylor_frame.jnl	
taylor_label.jnl	
taylor_plot.jnl	
taylor_polymark.jnl	
taylor_rgraticule.jnl	Plot graticules (lines) at specified radius (relative to standard deviation)
taylor_rmscircles.jnl	Plot RMS semicircles centered around data ref point on bottom axis
taylor_wtarea.jnl	Computes statistics for Taylor diagram (weights by area array defined a priori)
TernaryDiagram.jnl	
test_legend.jnl	
test.jnl	a dummy test journal file to execute
tgridlines.jnl	plot over the vertical grid of a plot
tics.jnl	Reset the plot style to use axis tics (see GO graticule)
tlandr.jnl	an outdated routine equivalent to "GO land"
topo_palette_demo.jnl	
topographic_relief_demo.jnl	reviews coarse subsets of the ETOPO (Equator to Poles) data set
trackplot_demo.jnl	demonstrate usage of trackplot jnl file
trackplot.jnl	Plot values of a variable alongside a plotted track
try_centered_palette.jnl	4 plots to illustrate "split at zero" color bar densities
try_palette.jnl	View 4 plots to illustrate color bar densities
try_pattern.jnl	View 4 plots to illustrate patterns
ts_frequency.jnl	produce a TS frequency "histogram"
TS_section.jnl	
tutorial.jnl	A brief tutorial introduction to FERRET
two_dee_plot_of_1d.jnl	Illustrates FILLing under a 1D plot (e.g. ocean bottom)
unbold.jnl	restore PLOT+ & FERRET to normal styling following "GO bold"
unit_square.jnl	set up the unit square as the default for abstract variables
unlabel.jnl	
unmagnify.jnl	restore the plot origin and axis lengths to default values
var_n.jnl	Correct "GO variance" definitions by n/n-1 factor
variance-ellipses-bold.jnl	
variance.jnl	define FERRET variables for covariance and correlation
vector_demo.jnl	Demonstration of various vector plotting options
vertical_section.jnl	Create an arbitrary 2D vertical section from a 3D field
vfland.jnl	Draw a filled vertical bathymetry slice in the x or y direction
viewports_demo.jnl	a quick (cutsie) demo using line plots and viewports
vis5d_append.jnl	Set filename and write a vis5d file with up to 8 variables.
vis5d_start.jnl	Set filename and start vis5d .
vis5d_write.jnl	Write up to 8 Ferret variables to a Vis5D file.
vland.jnl	Draw a vertical bathymetry line showing a slice in the x or y direction
water_vector_stack.jnl	
white.jnl	set video background to white, foreground to black
wind_barbs.jnl	illustrates how to draw wind barbs "from scratch" using ALINE

Skript Name	Erläuterung
wire_frame_demo.jnl	demonstrates how to create a 3D wire frame plot
wv5d_append.jnl	Set filename and write a vis5d file with up to 8 variables.
wv5d.jnl	Set filename and write a vis5d file with up to 8 variables.
xgridlines.jnl	plot over the vertical grid of a plot
ygridlines.jnl	plot over the horizontal grid of a plot

E Internal Functions

Funktion	Erläuterung
EXP(X)	exponential e(X)
LOG(X)	base 10 log(X)
MAX(A,B)	point-by-point greater of A and B
MIN(A,B)	point-by-point lesser of A and B
INT(X)	truncate to integer
ABS(X)	absolute value
SIN(theta)	theta: angle (radians)
COS(theta)	theta: angle (radians)
TAN(theta)	theta: angle (radians)
LN(X)	natural logarithm(X)
MOD(A,B)	modulo A using base B
MISSING(A,B)	substitute B where A is missing
IGNORE0(X)	substitute missing value for zeros
ATAN(X)	arctan(X) in radians
ATAN2(A,B)	arctan(A/B) in radians, -pi<result<=pi A: if A>0 result>0, if A=0 B determines result B: if A=0&B>0 res=0, if A=0&B<0 res=pi, if B=0 ABS(res)=pi/2
ASIN(X)	arcsin(X) in radians X: ABS(X) must be less than or equal to 1
ACOS(X)	arccos(X) in radians X: ABS(X) must be less than or equal to 1
RANDU(A)	random uniform [0,1] seeded from 1st value of A A: field of random values will have shape of A
RANDN(A)	random normal seeded from 1st value of A A: field of random values will have shape of A
RHO_UN(salt,temp,p)	UNESCO state equation (density) for ocean H2O (kg/m^3) salt: salinity (psu) temp: temperature (deg. C) p: reference pressure (decibars)
THETA_FO(salt,temp,p,ref)	Fofonoff (1977) potential temperature salt: salinity (psu) temp: temperature (deg. C) p: pressure (decibars) ref: reference pressure (decibars)
DAYS1900(year,month,day)	days elapsed since 1-Jan-1900 (standard calendar)
RANDU2(A,ISEED)	random uniform [0,1), Alternate algorithm A: field of random values will have shape of A ISEED: -1=sys clock, 0=continue w/ previous seed, N>0 user-defined seed
RANDN2(A,ISEED)	random normal, Alternate algorithm A: field of random values will have shape of A ISEED: -1=sys clock, 0=continue w/ previous seed, N>0 user-defined seed
XSEQUENCE(VAR)	unravel grid to a line in X
ECHO(STR,NUM)	echo passed string (test routine) STR: input text (STRING) NUM: numerical variable (anything)
RESHAPE(A,B)	reshape A to grid of B A: data to be reshaped B: destination grid
ZAXREPLACE(V,ZVALS,ZAX)	regrid V onto Z axis of ZAX based upon Z values in ZVALS using linear interpolation

Funktion	Erläuterung
	V: variable on native Z axis ZVALS: Z-value field corresponding to data points of V ZAX: variable with desired Z (depth) axis points
YSEQUENCE(VAR)	unravel grid to a line in Y
ZSEQUENCE(VAR)	unravel grid to a line in Z
TSEQUENCE(VAR)	unravel grid to a line in T
ESEQUENCE(VAR)	unravel grid to a line in E
FSEQUENCE(VAR)	unravel grid to a line in F
SAMPLEI(TO_BE_SAMPLED,X_INDICES)	sample a field at a list of X indices TO_BE_SAMPLED: data to sample at list of X indices supplied X_INDICES: list of X indices at which to sample
SAMPLEJ(TO_BE_SAMPLED,Y_INDICES)	sample a field at a list of Y indices TO_BE_SAMPLED: data to sample at list of Y indices supplied Y_INDICES: list of Y indices at which to sample
SAMPLEK(TO_BE_SAMPLED,Z_INDICES)	sample a field at a list of Z indices TO_BE_SAMPLED: data to sample at list of Z indices supplied Z_INDICES: list of Z indices at which to sample
SAMPLEL(TO_BE_SAMPLED,T_INDICES)	sample a field at a list of T indices TO_BE_SAMPLED: data to sample at list of T indices supplied T_INDICES: list of T indices at which to sample
SAMPLEM(TO_BE_SAMPLED,E_INDICES)	sample a field at a list of E indices TO_BE_SAMPLED: data to sample at list of E indices supplied E_INDICES: list of E indices at which to sample
SAMPLEN(TO_BE_SAMPLED,F_INDICES)	sample a field at a list of F indices TO_BE_SAMPLED: data to sample at list of F indices supplied F_INDICES: list of F indices at which to sample
SPAWN(STR)	execute a system command STR: Unix command string (STRING)
STRCMP(STR1,STR2)	compare strings: +, 0, or - for str1 >, =, or < str2 rspctvly STR1: string array 1 (STRING) STR2: string array 2 (STRING)
STRLEN(STR)	determine string length STR: string array 1 (STRING)
UPCASE(STR)	upper case entire string STR: string array 1 (STRING)
STRINDEX(STR1,SUBSTR)	Return start character position of substring in string STR1: string array 1 (STRING) SUBSTR: string array 2 (STRING)
STRRINDEX(STR1,SUBSTR)	Return last character position of substring in string STR1: string array 1 (STRING) SUBSTR: string array 2 (STRING)
DNCASE(STR)	make entire string lower case STR: string array 1 (STRING)
STRCAT(STR1,STR2)	concatenate two strings STR1: string array 1 (STRING) STR2: string array 2 (STRING)
SUBSTRING(STR,OFFSET,LENGTH)	Return a substring

Funktion	Erläuterung
	STR: string array (STRING)
	OFFSET: start position
	LENGTH: number of chars
`STRFLOAT(STR)`	Return float value from character string
	STR: string array 1 (STRING)

F Ferret Demo Dateien

coads_climatology.cdf

```
ncdump -h coads_climatology.cdf
netcdf coads_climatology {
dimensions:
        COADSX = 180 ;
        COADSY = 90 ;
        TIME = UNLIMITED ; // (12 currently)
variables:
        double COADSX(COADSX) ;
                COADSX:units = "degrees_east" ;
                COADSX:modulo = " " ;
                COADSX:point_spacing = "even" ;
        double COADSY(COADSY) ;
                COADSY:units = "degrees_north" ;
                COADSY:point_spacing = "even" ;
        double TIME(TIME) ;
                TIME:units = "hour since 0000-01-01 00:00:00" ;
                TIME:time_origin = "1-JAN-0000 00:00:00" ;
                TIME:modulo = " " ;
        float SST(TIME, COADSY, COADSX) ;
                SST:missing_value = -1.e+34f ;
                SST:_FillValue = -1.e+34f ;
                SST:long_name = "SEA SURFACE TEMPERATURE" ;
                SST:history = "From coads_climatology" ;
                SST:units = "Deg C" ;
        float AIRT(TIME, COADSY, COADSX) ;
                AIRT:missing_value = -1.e+34f ;
                AIRT:_FillValue = -1.e+34f ;
                AIRT:long_name = "AIR TEMPERATURE" ;
                AIRT:history = "From coads_climatology" ;
                AIRT:units = "DEG C" ;
        float SPEH(TIME, COADSY, COADSX) ;
                SPEH:missing_value = -1.e+34f ;
                SPEH:_FillValue = -1.e+34f ;
                SPEH:long_name = "SPECIFIC HUMIDITY" ;
                SPEH:history = "From coads_climatology" ;
                SPEH:units = "G/KG" ;
        float WSPD(TIME, COADSY, COADSX) ;
                WSPD:missing_value = -1.e+34f ;
                WSPD:_FillValue = -1.e+34f ;
                WSPD:long_name = "WIND SPEED" ;
                WSPD:history = "From coads_climatology" ;
                WSPD:units = "M/S" ;
        float UWND(TIME, COADSY, COADSX) ;
                UWND:missing_value = -1.e+34f ;
                UWND:_FillValue = -1.e+34f ;
                UWND:long_name = "ZONAL WIND" ;
                UWND:history = "From coads_climatology" ;
                UWND:units = "M/S" ;
        float VWND(TIME, COADSY, COADSX) ;
                VWND:missing_value = -1.e+34f ;
                VWND:_FillValue = -1.e+34f ;
                VWND:long_name = "MERIDIONAL WIND" ;
                VWND:history = "From coads_climatology" ;
                VWND:units = "M/S" ;
        float SLP(TIME, COADSY, COADSX) ;
                SLP:missing_value = -1.e+34f ;
                SLP:_FillValue = -1.e+34f ;
                SLP:long_name = "SEA LEVEL PRESSURE" ;
                SLP:history = "From coads_climatology" ;
                SLP:units = "MB" ;

// global attributes:
                :history = "FERRET V4.45 (GUI) 22-May-97" ;
}
```

esku_heat_budget.cdf

```
ncdump -h esku_heat_budget.cdf
netcdf esku_heat_budget {
dimensions:
        ESKUX = 72 ;
        ESKUY = 46 ;
        ESKUYedges = 47 ;
        TIME = UNLIMITED ; // (12 currently)
variables:
        double ESKUX(ESKUX) ;
                ESKUX:units = "degrees_east" ;
                ESKUX:modulo = " " ;
                ESKUX:point_spacing = "even" ;
        double ESKUY(ESKUY) ;
                ESKUY:units = "degrees_north" ;
                ESKUY:point_spacing = "uneven" ;
                ESKUY:edges = "ESKUYedges" ;
        double ESKUYedges(ESKUYedges) ;
                ESKUYedges:edges = " " ;
        double TIME(TIME) ;
                TIME:units = "hour since 0000-01-01 00:00:00" ;
                TIME:time_origin = "1-JAN-0000 00:00:00" ;
                TIME:modulo = " " ;
        float SPD(TIME, ESKUY, ESKUX) ;
                SPD:missing_value = 1.e+34f ;
                SPD:_FillValue = 1.e+34f ;
                SPD:long_name = "SURFACE WIND SPEED" ;
                SPD:history = "From esku_heat_budget" ;
                SPD:units = "M/S" ;
        float SST(TIME, ESKUY, ESKUX) ;
                SST:missing_value = 1.e+34f ;
                SST:_FillValue = 1.e+34f ;
                SST:long_name = "SEA SURFACE TEMPERATURE" ;
                SST:history = "From esku_heat_budget" ;
                SST:units = "DEG C" ;
        float SAT(TIME, ESKUY, ESKUX) ;
                SAT:missing_value = 1.e+34f ;
                SAT:_FillValue = 1.e+34f ;
                SAT:long_name = "SEA-AIR TEMPERATURE DIFFERENCE" ;
                SAT:history = "From esku_heat_budget" ;
                SAT:units = "DEG C" ;
        float AT(TIME, ESKUY, ESKUX) ;
                AT:missing_value = 1.e+34f ;
                AT:_FillValue = 1.e+34f ;
                AT:long_name = "AIR TEMPERATURE" ;
                AT:history = "From esku_heat_budget" ;
                AT:units = "DEG C" ;
        float AH(TIME, ESKUY, ESKUX) ;
                AH:missing_value = 1.e+34f ;
                AH:_FillValue = 1.e+34f ;
                AH:long_name = "SPECIFIC HUMIDITY" ;
                AH:history = "From esku_heat_budget" ;
                AH:units = "GR/KG" ;
        float SAH(TIME, ESKUY, ESKUX) ;
                SAH:missing_value = 1.e+34f ;
                SAH:_FillValue = 1.e+34f ;
                SAH:long_name = "SEA-AIR SPECIFIC HUMIDITY DIFFERENCE" ;
                SAH:history = "From esku_heat_budget" ;
                SAH:units = "GR/KG" ;
        float CLD(TIME, ESKUY, ESKUX) ;
                CLD:missing_value = 1.e+34f ;
                CLD:_FillValue = 1.e+34f ;
                CLD:long_name = "CLOUDINESS" ;
                CLD:history = "From esku_heat_budget" ;
                CLD:units = "FRACTION OF SKY COVER" ;
        float SLP(TIME, ESKUY, ESKUX) ;
                SLP:missing_value = 1.e+34f ;
                SLP:_FillValue = 1.e+34f ;
                SLP:long_name = "SEA LEVEL PRESSURE" ;
                SLP:history = "From esku_heat_budget" ;
                SLP:units = "MB" ;
        float FSR(TIME, ESKUY, ESKUX) ;
```

```
        FSR:missing_value = 1.e+34f ;
        FSR:_FillValue = 1.e+34f ;
        FSR:long_name = "AVAILABLE SOLAR RADIATION" ;
        FSR:history = "From esku_heat_budget" ;
        FSR:units = "W/M2" ;
float FUL(TIME, ESKUY, ESKUX) ;
        FUL:missing_value = 1.e+34f ;
        FUL:_FillValue = 1.e+34f ;
        FUL:long_name = "NET UPWARD LONGWAVE FLUX" ;
        FUL:history = "From esku_heat_budget" ;
        FUL:units = "W/M2" ;
float FDR(TIME, ESKUY, ESKUX) ;
        FDR:missing_value = 1.e+34f ;
        FDR:_FillValue = 1.e+34f ;
        FDR:long_name = "NET DOWNWARD RADIATIVE FLUX" ;
        FDR:history = "From esku_heat_budget" ;
        FDR:units = "W/M2" ;
float FLH(TIME, ESKUY, ESKUX) ;
        FLH:missing_value = 1.e+34f ;
        FLH:_FillValue = 1.e+34f ;
        FLH:long_name = "LATENT HEAT FLUX" ;
        FLH:history = "From esku_heat_budget" ;
        FLH:units = "W/M2" ;
float FSH(TIME, ESKUY, ESKUX) ;
        FSH:missing_value = 1.e+34f ;
        FSH:_FillValue = 1.e+34f ;
        FSH:long_name = "SENSIBLE HEAT FLUX" ;
        FSH:history = "From esku_heat_budget" ;
        FSH:units = "W/M2" ;
float FDH(TIME, ESKUY, ESKUX) ;
        FDH:missing_value = 1.e+34f ;
        FDH:_FillValue = 1.e+34f ;
        FDH:long_name = "NET DOWNWARD HEAT FLUX" ;
        FDH:history = "From esku_heat_budget" ;
        FDH:units = "W/M2" ;
float KSPD(TIME, ESKUY, ESKUX) ;
        KSPD:missing_value = 1.e+34f ;
        KSPD:_FillValue = 1.e+34f ;
        KSPD:long_name = "DATA DENSITY OF SURFACE WIND SPEED" ;
        KSPD:history = "From esku_heat_budget" ;
        KSPD:units = "LOG10 #OBS" ;
float KSST(TIME, ESKUY, ESKUX) ;
        KSST:missing_value = 1.e+34f ;
        KSST:_FillValue = 1.e+34f ;
        KSST:long_name = "DATA DENSITY OF SST" ;
        KSST:history = "From esku_heat_budget" ;
        KSST:units = "LOG10 #OBS" ;
float KSAT(TIME, ESKUY, ESKUX) ;
        KSAT:missing_value = 1.e+34f ;
        KSAT:_FillValue = 1.e+34f ;
        KSAT:long_name = "DATA DENSITY OF SEA-AIR TEMPERATURE DIFF" ;
        KSAT:history = "From esku_heat_budget" ;
        KSAT:units = "LOG10 #OBS" ;
float KAT(TIME, ESKUY, ESKUX) ;
        KAT:missing_value = 1.e+34f ;
        KAT:_FillValue = 1.e+34f ;
        KAT:long_name = "DATA DENSITY OF AIR TEMPERATURE" ;
        KAT:history = "From esku_heat_budget" ;
        KAT:units = "LOG10 #OBS" ;
float KAH(TIME, ESKUY, ESKUX) ;
        KAH:missing_value = 1.e+34f ;
        KAH:_FillValue = 1.e+34f ;
        KAH:long_name = "DATA DENSITY OF SPECIFIC HUMIDITY" ;
        KAH:history = "From esku_heat_budget" ;
        KAH:units = "LOG10 #OBS" ;
float KSAH(TIME, ESKUY, ESKUX) ;
        KSAH:missing_value = 1.e+34f ;
        KSAH:_FillValue = 1.e+34f ;
        KSAH:long_name = "DATA DENSITY OF SEA-AIR SPECIFIC HUMIDIT" ;
        KSAH:history = "From esku_heat_budget" ;
        KSAH:units = "LOG10 #OBS" ;
float KSLP(TIME, ESKUY, ESKUX) ;
```

```
                KSLP:missing_value = 1.e+34f ;
                KSLP:_FillValue = 1.e+34f ;
                KSLP:long_name = "DATA DENSITY OF SEA LEVEL PRESSURE" ;
                KSLP:history = "From esku_heat_budget" ;
                KSLP:units = "LOG10 #OBS" ;
        float KFUL(TIME, ESKUY, ESKUX) ;
                KFUL:missing_value = 1.e+34f ;
                KFUL:_FillValue = 1.e+34f ;
                KFUL:long_name = "DATA DENSITY OF NET UPWARD LONGWAVE FLUX" ;
                KFUL:history = "From esku_heat_budget" ;
                KFUL:units = "LOG10 #OBS" ;
        float KFLH(TIME, ESKUY, ESKUX) ;
                KFLH:missing_value = 1.e+34f ;
                KFLH:_FillValue = 1.e+34f ;
                KFLH:long_name = "DATA DENSITY OF LATENT HEAT FLUX" ;
                KFLH:history = "From esku_heat_budget" ;
                KFLH:units = "LOG10 #OBS" ;
        float KFSH(TIME, ESKUY, ESKUX) ;
                KFSH:missing_value = 1.e+34f ;
                KFSH:_FillValue = 1.e+34f ;
                KFSH:long_name = "DATA DENSITY OF SENSIBLE HEAT FLUX" ;
                KFSH:history = "From esku_heat_budget" ;
                KFSH:units = "LOG10 #OBS" ;
        float KFDH(TIME, ESKUY, ESKUX) ;
                KFDH:missing_value = 1.e+34f ;
                KFDH:_FillValue = 1.e+34f ;
                KFDH:long_name = "DATA DENSITY OF NET DOWNWARD HEAT FLUX" ;
                KFDH:history = "From esku_heat_budget" ;
                KFDH:units = "LOG10 #OBS" ;

// global attributes:
                :history = "FERRET V4.45 (GUI) 22-May-97" ;
}
```

etopo120.cdf

```
ncdump -h etopo120.cdf
netcdf etopo120 {
dimensions:
        ETOPO120X = 180 ;
        ETOPO120Y = 90 ;
variables:
        double ETOPO120X(ETOPO120X) ;
                ETOPO120X:units = "degrees_east" ;
                ETOPO120X:modulo = " " ;
                ETOPO120X:point_spacing = "even" ;
        double ETOPO120Y(ETOPO120Y) ;
                ETOPO120Y:units = "degrees_north" ;
                ETOPO120Y:point_spacing = "even" ;
        float ROSE(ETOPO120Y, ETOPO120X) ;
                ROSE:missing_value = -1.e+34f ;
                ROSE:_FillValue = -1.e+34f ;
                ROSE:long_name = "RELIEF OF THE SURFACE OF THE EARTH" ;
                ROSE:history = "From etopo120" ;
                ROSE:units = "METERS" ;

// global attributes:
                :history = "FERRET V4.45 (GUI) 22-May-97" ;
}
```

etopo20.cdf

```
ncdump -h etopo20.cdf
netcdf etopo20 {
dimensions:
        ETOPO20X1_1081 = 1081 ;
        ETOPO20Y = 540 ;
variables:
        double ETOPO20X1_1081(ETOPO20X1_1081) ;
```

```
                ETOPO20X1_1081:units = "degrees_east" ;
                ETOPO20X1_1081:modulo = " " ;
                ETOPO20X1_1081:point_spacing = "even" ;
        double ETOPO20Y(ETOPO20Y) ;
                ETOPO20Y:units = "degrees_north" ;
                ETOPO20Y:point_spacing = "even" ;
        float ROSE(ETOPO20Y, ETOPO20X1_1081) ;
                ROSE:missing_value = -1.e+34f ;
                ROSE:_FillValue = -1.e+34f ;
                ROSE:long_name = "RELIEF OF THE SURFACE OF THE EARTH" ;
                ROSE:history = "From etopo20" ;
                ROSE:units = "METERS" ;

// global attributes:
                :history = "FERRET V4.45 (GUI) 22-May-97" ;
}
```

etopo40.cdf

```
ncdump -h etopo40.cdf
netcdf etopo40 {
dimensions:
        ETOPO40X = 540 ;
        ETOPO40Y = 270 ;
variables:
        double ETOPO40X(ETOPO40X) ;
                ETOPO40X:units = "degrees_east" ;
                ETOPO40X:modulo = " " ;
                ETOPO40X:point_spacing = "even" ;
        double ETOPO40Y(ETOPO40Y) ;
                ETOPO40Y:units = "degrees_north" ;
                ETOPO40Y:point_spacing = "even" ;
        float ROSE(ETOPO40Y, ETOPO40X) ;
                ROSE:missing_value = -1.e+34f ;
                ROSE:_FillValue = -1.e+34f ;
                ROSE:long_name = "RELIEF OF THE SURFACE OF THE EARTH" ;
                ROSE:history = "From etopo40" ;
                ROSE:units = "METERS" ;

// global attributes:
                :history = "FERRET V4.91 (GUI) 12-Jun-98" ;
}
```

etopo5.cdf

```
ncdump -h etopo5.cdf
netcdf etopo5 {
dimensions:
        ETOPO05_X = 4320 ;
        ETOPO05_Y = 2161 ;
variables:
        double ETOPO05_X(ETOPO05_X) ;
                ETOPO05_X:modulo = " " ;
                ETOPO05_X:point_spacing = "even" ;
                ETOPO05_X:units = "degrees_east" ;
        double ETOPO05_Y(ETOPO05_Y) ;
                ETOPO05_Y:point_spacing = "even" ;
                ETOPO05_Y:units = "degrees_north" ;
        float ROSE(ETOPO05_Y, ETOPO05_X) ;
                ROSE:missing_value = -1.e+34f ;
                ROSE:_FillValue = -1.e+34f ;
                ROSE:long_name = "Relief Of the Surface of the Earth" ;
                ROSE:history = "From worldbath.nc" ;
                ROSE:units = "meters" ;

// global attributes:
                :history = "FERRET V5.22   27-Apr-01, from IRI/LDEO worldbath.nc" ;
                :IRI_LDEO_note = "updated 27 Feb 1998 from NGDC CD-ROM 29 April 1993"
                    ;
}
```

etopo60.cdf

```
ncdump -h etopo60.cdf
netcdf etopo60 {
dimensions:
        ETOPO60X = 360 ;
        ETOPO60Y = 180 ;
variables:
        double ETOPO60X(ETOPO60X) ;
                ETOPO60X:units = "degrees_east" ;
                ETOPO60X:modulo = " " ;
                ETOPO60X:point_spacing = "even" ;
        double ETOPO60Y(ETOPO60Y) ;
                ETOPO60Y:units = "degrees_north" ;
                ETOPO60Y:point_spacing = "even" ;
        float ROSE(ETOPO60Y, ETOPO60X) ;
                ROSE:missing_value = -1.e+34f ;
                ROSE:_FillValue = -1.e+34f ;
                ROSE:long_name = "RELIEF OF THE SURFACE OF THE EARTH" ;
                ROSE:history = "From etopo60" ;
                ROSE:units = "METERS" ;

// global attributes:
                :history = "FERRET V4.45 (GUI) 22-May-97" ;
}
```

levitus_climatology.cdf

```
ncdump -h levitus_climatology.cdf
netcdf levitus_climatology {
dimensions:
        XAXLEVITR = 360 ;
        YAXLEVITR = 180 ;
        ZAXLEVITR = 20 ;
        ZAXLEVITRedges = 21 ;
variables:
        double XAXLEVITR(XAXLEVITR) ;
                XAXLEVITR:units = "degrees_east" ;
                XAXLEVITR:modulo = " " ;
                XAXLEVITR:point_spacing = "even" ;
        double YAXLEVITR(YAXLEVITR) ;
                YAXLEVITR:units = "degrees_north" ;
                YAXLEVITR:point_spacing = "even" ;
        double ZAXLEVITR(ZAXLEVITR) ;
                ZAXLEVITR:units = "METERS" ;
                ZAXLEVITR:positive = "down" ;
                ZAXLEVITR:point_spacing = "uneven" ;
                ZAXLEVITR:edges = "ZAXLEVITRedges" ;
        double ZAXLEVITRedges(ZAXLEVITRedges) ;
                ZAXLEVITRedges:edges = " " ;
        float TEMP(ZAXLEVITR, YAXLEVITR, XAXLEVITR) ;
                TEMP:missing_value = -1.e+10f ;
                TEMP:_FillValue = -1.e+10f ;
                TEMP:long_name = "TEMPERATURE" ;
                TEMP:history = "From levitus_climatology" ;
                TEMP:units = "DEG C" ;
        float SALT(ZAXLEVITR, YAXLEVITR, XAXLEVITR) ;
                SALT:missing_value = -1.e+10f ;
                SALT:_FillValue = -1.e+10f ;
                SALT:long_name = "SALINITY" ;
                SALT:history = "From levitus_climatology" ;
                SALT:units = "PPT" ;

// global attributes:
                :history = "FERRET V4.45 (GUI) 22-May-97" ;
}
```

monthly_navy_winds.cdf

```
ncdump -h monthly_navy_winds.cdf
netcdf monthly_navy_winds {
dimensions:
        FNOCX = 144 ;
        FNOCY = 73 ;
        TIME = UNLIMITED ; // (132 currently)
variables:
        double FNOCX(FNOCX) ;
                FNOCX:units = "degrees_east" ;
                FNOCX:modulo = " " ;
                FNOCX:point_spacing = "even" ;
        double FNOCY(FNOCY) ;
                FNOCY:units = "degrees_north" ;
                FNOCY:point_spacing = "even" ;
        double TIME(TIME) ;
                TIME:units = "hour since 1980-01-14 14:00:00" ;
                TIME:time_origin = "14-JAN-1980 14:00:00" ;
        float UWND(TIME, FNOCY, FNOCX) ;
                UWND:missing_value = -99.9f ;
                UWND:_FillValue = -99.9f ;
                UWND:long_name = "ZONAL WIND" ;
                UWND:history = "From monthly_navy_winds" ;
                UWND:units = "M/S" ;
        float VWND(TIME, FNOCY, FNOCX) ;
                VWND:missing_value = -99.9f ;
                VWND:_FillValue = -99.9f ;
                VWND:long_name = "MERIDIONAL WIND" ;
                VWND:history = "From monthly_navy_winds" ;
                VWND:units = "M/S" ;

// global attributes:
                :history = "FERRET V4.45 (GUI) 22-May-97" ;
}
```

climatological_axes.cdf

```
netcdf climatological_axes {
dimensions:
        MONTH_IRREG = 12 ;
        MONTH_IRREGedges = 13 ;
        MONTH_REG = 12 ;
        SEASONAL_REG = 4 ;
        MONTH_GREGORIAN = 12 ;
        MONTH_GREGORIAN_EDGES = 13 ;
        MONTH_NOLEAP = 12 ;
        MONTH_NOLEAP_EDGES = 13 ;
        MONTH_360_DAY = 12 ;
        MONTH_ALL_LEAP = 12 ;
        MONTH_ALL_LEAP_EDGES = 13 ;
variables:
        double MONTH_IRREG(MONTH_IRREG) ;
                MONTH_IRREG:units = "DAYS since 0000-01-01 00:00:00" ;
                MONTH_IRREG:time_origin = "1-JAN-0000" ;
                MONTH_IRREG:modulo = " " ;
                MONTH_IRREG:edges = "MONTH_IRREGedges" ;
        double MONTH_IRREGedges(MONTH_IRREGedges) ;
                MONTH_IRREGedges:edges = " " ;
        double MONTH_REG(MONTH_REG) ;
                MONTH_REG:units = "hour since 0000-01-01 00:00:00" ;
                MONTH_REG:MONTH_REG_origin = "1-JAN-0000 00:00:00" ;
                MONTH_REG:modulo = " " ;
        double SEASONAL_REG(SEASONAL_REG) ;
                SEASONAL_REG:units = "HOURS since 0000-01-01 00:00:00" ;
                SEASONAL_REG:time_origin = "01-JAN-0000 00:00:00" ;
                SEASONAL_REG:modulo = " " ;
        double MONTH_GREGORIAN(MONTH_GREGORIAN) ;
                MONTH_GREGORIAN:units = "days since 0000-01-01" ;
                MONTH_GREGORIAN:time_origin = "1-JAN-0000" ;
                MONTH_GREGORIAN:modulo = " " ;
```

```
                MONTH_GREGORIAN:edges = "MONTH_GREGORIAN_EDGES" ;
                MONTH_GREGORIAN:calendar = "gregorian" ;
        double MONTH_GREGORIAN_EDGES(MONTH_GREGORIAN_EDGES) ;
                MONTH_GREGORIAN_EDGES:edges = " " ;
        double MONTH_NOLEAP(MONTH_NOLEAP) ;
                MONTH_NOLEAP:units = "days since 0000-01-01" ;
                MONTH_NOLEAP:time_origin = "1-JAN-0000" ;
                MONTH_NOLEAP:modulo = " " ;
                MONTH_NOLEAP:edges = "MONTH_NOLEAP_EDGES" ;
                MONTH_NOLEAP:calendar = "noleap" ;
        double MONTH_NOLEAP_EDGES(MONTH_NOLEAP_EDGES) ;
                MONTH_NOLEAP_EDGES:edges = " " ;
        double MONTH_360_DAY(MONTH_360_DAY) ;
                MONTH_360_DAY:units = "days since 0000-01-01" ;
                MONTH_360_DAY:time_origin = "1-JAN-0000" ;
                MONTH_360_DAY:modulo = " " ;
                MONTH_360_DAY:point_spacing = "even" ;
                MONTH_360_DAY:calendar = "360_day" ;
        double MONTH_ALL_LEAP(MONTH_ALL_LEAP) ;
                MONTH_ALL_LEAP:units = "days since 0000-01-01" ;
                MONTH_ALL_LEAP:time_origin = "1-JAN-0000" ;
                MONTH_ALL_LEAP:modulo = " " ;
                MONTH_ALL_LEAP:edges = "MONTH_ALL_LEAP_EDGES" ;
                MONTH_ALL_LEAP:calendar = "all_leap" ;
        double MONTH_ALL_LEAP_EDGES(MONTH_ALL_LEAP_EDGES) ;
                MONTH_ALL_LEAP_EDGES:edges = " " ;

// global attributes:
                :history = "Axes with years of exactly 365.2425 days" ;
                :message = "Climatological axes SEASONAL_REG, MONTH_REG, MONTH_IRREG,
                    MONTH_GREGORIAN, MONTH_NOLEAP, MONTH_360_DAY, MONTH_ALL_LEAP
                    defined" ;
}
```

polydata.cdf

```
ncdump -h polydata.cdf
netcdf polydata {
dimensions:
        XAX1_37 = 37 ;
variables:
        double XAX1_37(XAX1_37) ;
                XAX1_37:point_spacing = "even" ;
        float LON(XAX1_37) ;
                LON:missing_value = -1.e+34f ;
                LON:_FillValue = -1.e+34f ;
                LON:long_name = "LON" ;
                LON:history = "From temp_small.dat" ;
        float LAT(XAX1_37) ;
                LAT:missing_value = -1.e+34f ;
                LAT:_FillValue = -1.e+34f ;
                LAT:long_name = "LAT" ;
                LAT:history = "From temp_small.dat" ;
        float SST(XAX1_37) ;
                SST:missing_value = -1.e+34f ;
                SST:_FillValue = -1.e+34f ;
                SST:long_name = "SST" ;
                SST:history = "From temp_small.dat" ;

// global attributes:
                :history = "FERRET V5.00 (V500beta1.0) 18-May-99" ;
}
```

G CSV Beispiel Dateien

Listing 1: data1.csv

```
   7.396
   7.079
   7.019
   6.969
   7.462
   8.950
  10.443
  10.946
  10.416
   9.088
   8.177
   7.474
```

Listing 2: data2.csv

```
Monat,  SST
Januar,   7.396
Februar,   7.079
März,   7.019
April,   6.969
Mai,   7.462
Juni,   8.950
Juli,  10.443
August,  10.946
September,  10.416
Oktober,   9.088
November,   8.177
Dezember,   7.474
```

Listing 3: data2a.csv

```
Monat  SST
Januar   7.396
Februar   7.079
März   7.019
April   6.969
Mai   7.462
Juni   8.950
Juli  10.443
August  10.946
September  10.416
Oktober   9.088
November   8.177
Dezember   7.474
```

Listing 4: data3.csv

```
 Datum     SST    AIRT
16-JAN   7.396   4.137
15-FEB   7.079   4.236
17-MAR   7.019   4.332
16-APR   6.969   5.956
16-MAY   7.462   6.987
16-JUN   8.950   8.890
16-JUL  10.443  10.188
16-AUG  10.946  10.728
15-SEP  10.416   9.847
16-OCT   9.088   7.783
15-NOV   8.177   5.965
16-DEC   7.474   5.049
```

Listing 5: data4.csv

```
  Zeit     SST    AIRT
   366.  7.396   4.137
  1096.  7.079   4.236
  1827.  7.019   4.332
  2557.  6.969   5.956
  3288.  7.462   6.987
  4018.  8.950   8.890
  4749. 10.443  10.188
  5479. 10.946  10.728
  6210. 10.416   9.847
  6940.  9.088   7.783
  7671.  8.177   5.965
  8401.  7.474   5.049
```