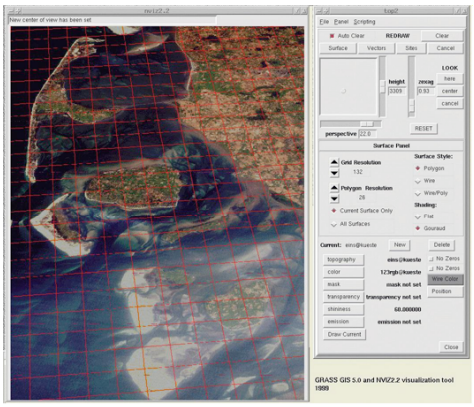
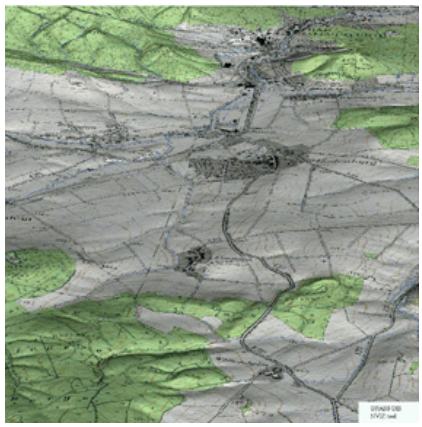
MANUAL DE GRASS

INTRODUCCIÓN A GRASS V.1.1





Autor: CARLOS CAMPILLO TORRES
Colaboradores: T. Buyolo, J. Cabezas, J. Martín, D. Patón

Área de Ecología, Departamento de Física, Universidad de Extremadura

COLABORA:

Junta de Extremadura
Consejería de Educación, Ciencia y Tecnología

INDICE

Capitulo I: Empezando a utilizar GRASS

Objetivos:

1.Iniciar una sesión de GRASS5

El programa de GRASS se inicia utilizando la orden grass5 en este caso ya que vamos a utilizar la versión 5.0. El programa nos mostrara un mensaje de entrada, dependiendo si es la primera vez que se utiliza o ya se ha utilizado, si no se ha utilizado nunca saldrá el letrero de Bienvenida y para continuar solo necesitamos darle a **<enter>**. Inicialmente saldrá una pantalla compuesta principalmente por tres datos:

- LOCATION: Es el nombre de la región o localidad con la que se va a trabajar, constituye una estructura de datos.
- MAPSET: Cada sesión de trabajo se asocia a un MAPSET, en este se guardan todos los mapas generados
- DATABASE: Es el directorio de trabajo, donde se almacenaran estas bases de datos.

En esta zona debemos tener cuidado con el teclado ya que si no conocemos los terminales de texto Unix podemos liarnos sobretodo al intentar modificar algo. Podemos utilizar una serie de teclas combinadas con la tecla **<ctrl.>** como son:

H retrocede un carácter

J avanza a la línea siguiente

K vuelve a la línea anterior

L avanza una carácter

Space avanza carácter o borra caracteres

Para confirmar los datos seleccionados debemos pulsar la tecla escape y enter. Si por el contrario queremos cancelar todo se utiliza la tecla **<ctrl.>** + **c**.

Vamos a ver un ejemplo: Existen diferentes localizaciones de ejemplo en la pagina web de grass, utilizaremos principalmente la de leics y la de spearfish. http://grass.itc.it/data.html

LOCATION: leics

MAPSET: carlos (cualquier nombre que queramos)

DATABASE: home/grass (donde hemos guardado la base de datos leics)

Si hemos indicado todo correctamente, teclemamos **<esc>** + **<enter>**, aparecerá una terminal (pantalla blanca donde escribir) y según versiones arrancara automáticamente un programa paralelo a grass llamado **tcltkgrass** que sirve para sustituir el modo texto por un entorno de menús más cómodo para alguien que no este acostumbrado a los comandos.

Fig.1.1. Barra de menús para grass5 con tcltkgrass

En algunas ocasiones y en función siempre de las versiones puede aparecer un menú gráfico de inicio de grass que nos solucione la partida a la hora de elegir:

Fig. 1.2. Menú inicio grass versión 5

Grass es un programa de gestión de SIG pensado en trabajar en un entorno multiusuario en una red de ordenadores. Por tanto cada usuario debe tener su propio entorno de trabajo, es decir se MAPSET a lo que los demás usuarios pueden entrar pero no manipular. Además de estos personales existe un MAPSET especial llamado PERMANENT que contiene todos aquellos mapas que se consideran acabados, que no van a ser modificados, con lo que ningún usuario podrá modificarlos.

Cuando seleccionamos una LOCATION o un MAPSET que no están definidos nos van inicialmente a preguntar acerca de sus características.

Por ultimo una vez terminada la sesión de trabajo el programa se abandona escribiendo con la orden **exit**, otra opción es la de pulsar **<ctrl.>** + **c** para anular cualquier orden, tras ello se obtiene un informe de los mapas creados, con la posibilidad de eliminarlos o mantenerlos, según convenga.

Capitulo II: Introducción a GRASS

Una vez que ya sabemos iniciar una sesión en grass iniciamos la sesión con la LOCATION: leics, que es una localización inglesa en el noroeste de Leicestershire, esta zona tiene una superficie de 12 km2, con diferentes mapas tanto raster, como vectoriales. En este caso vamos a utilizar este mapa como forma de iniciarnos en algunos comandos posteriormente iremos utilizando otras localizaciones como otra que posee grass en su pagina web que es spearfish y utilizaremos también alguna propia de nuestra zona.

En este caso la zona de estudio tiene menor importancia, para aprender a manejarnos.

Objetivos:

1.Identificar los distintos tipos de comandos

2.Explorar una base de datos para ver los mapas que tenemos

3.Conocer el manejo de los comandos

- **g.list**: listado de mapas
- **g.manual**: muestra el manual de los diferentes comandos
- **d.mon**: Activa una pantalla para ver mapas
- **d.rast**: Permite ver un mapa en formato raster
- **d.vect**: Permite ver un mapa en formato vectorial
- **d.sites**: Permite ver un mapa de sites
- **d.erase**: Borra el contenido del monitor activo

1.Identificar distintos comandos

Al trabajar con los entornos Unix de texto nos obliga inicialmente a conocer los comandos para poder ponerlos en la terminal, pero esto lo soluciona grass con su aplicación tcltkgrass, en este caso explicaremos el funcionamiento de los comandos tanto en modo texto(mas útil para usuarios mas expertos) y modo gráfico(ideal para noveles).

Los nombres de los módulos en grass están constituidos por una letra minúscula que hace referencia al tipo de comando seguido de un punto y el nombre del comando. Los comandos son:

- d.nombre comando - módulos de salida gráfica
- g.nombre comando - módulos de gestión de ficheros
- i.nombre comando – módulos para tratamientos de imágenes de satélite
- r.nombre comando – módulos para análisis de capas raster
- v.nombre comando – módulos para análisis de capas vectoriales
- s.nombre comando – módulos de procesamiento de capas sites
- m.nombre comando – módulos miscelanea
- p.nombre comando – módulos de creación cartográfica
- ps.nombre comando - módulos de creación cartográfica

+ también tenemos la opción de conocer los nombres de los comandos que posee grass para cada tipo de comando introduciendo la letra del comando seguido del punto y luego la damos a la tecla de tabulación.

* en determinados comandos tenemos las opciones especiales indicadas por un prefijo antes del nombre del comando como son:

- in – módulos de importación de otros formatos a grass
- out – módulos de exportación de grass a otros formatos
- surf – interpola superficies a partir de mapas de puntos o lineas
- what – permite conocer las características del mapa mostrado en el monitor gráfico pinchando con el ratón sobre el
- to – se utiliza principalmente para pasar de un formato a otro de raster a vectorial por ejemplo.

Todos los comando de grass tienen un manual de uso de estos que se pueden ver con el comando g.manual; como ejemplo podemos ver el manual del siguiente comando que vamos a ver que es g.list, así que solo hay que indicar GRASS:~> g.manual g.list

2.Explorar una base de datos para ver los mapas que tenemos

- **g.list** permite conocer el listado de los mapas que tenemos en nuestro MAPSET

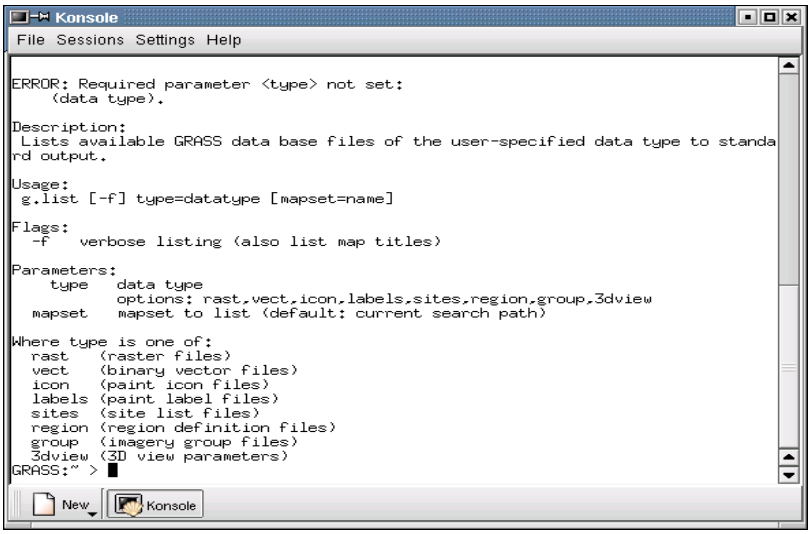


fig 2.1. Pantalla de GRASS en terminal modo texto comando **g.list -f**

además este comando lo tenemos en la opción de menús, si pinchamos en mapa y después en list

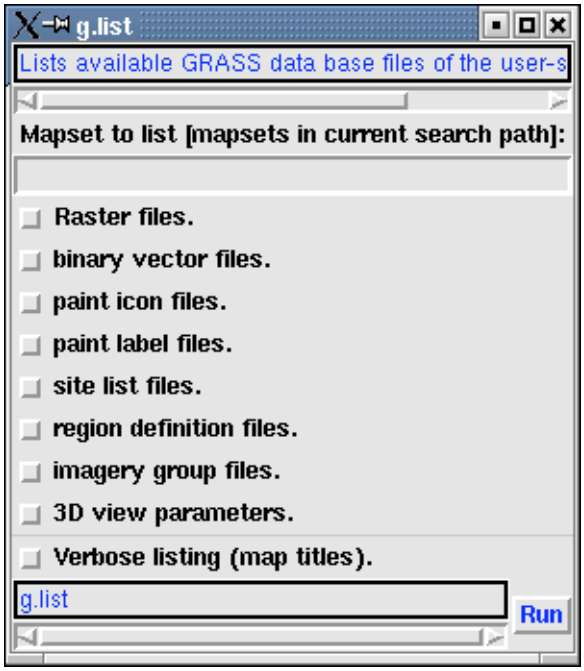


fig 2.2. Pantalla de GRASS en terminal modo gráfico comando **g.list**

aquí podemos ver las distintas posibilidades que tenemos a la hora de poder ver la base de datos de los mapas existentes, tanto mapas de formato raster como vectoriales, ficheros de iconos y etiquetas para crear cartografía, mapas de puntos, ficheros con las características de la región de trabajo, imágenes de satélite y formatos en tres dimensiones; así si ponemos por

ejemplo la opción de g.list raster o en formato texto **g.list rast**.

vemos que divide la información en dos tipos de mapas en formato raster los que tengo creados en mi MAPSET y los que existen en ese MAPSET pero en la sección PERMANENT, que se pueden ver pero no modificar.

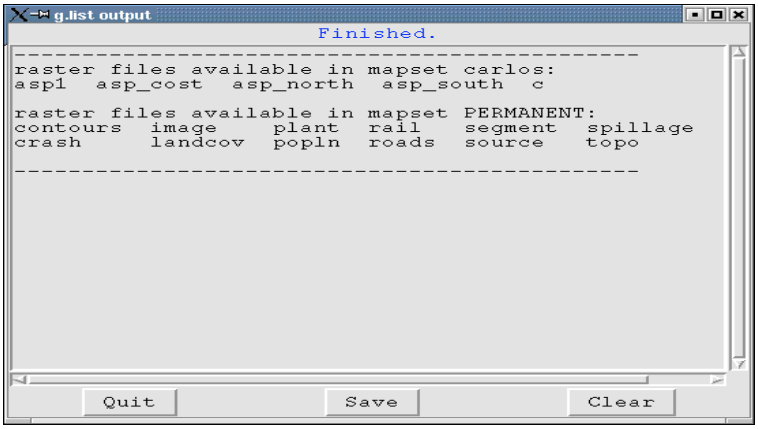


fig 2.3. Pantalla de mapas en nuestro MAPSET en formato raster

3.Identificar distintos comandos

- **g.list** de sobra conocido por el punto anterior que se vio una muestra
- **g.manual**, permite ver las indicaciones de cada comando todos están en ingles aunque ya existen algunos grandes colaboradores con este software como es D. F. Alonso Sarria y su equipo de la universidad de murcia.
- **d.mon**, este comando va a ser el primero que tecleemos o que seleccionemos nada más empezar una sesión de GRASS ya que es el encargado de lanzar una ventana gráfica donde podremos ver nuestros mapas. La sintaxis del comando es sencilla tanto si tecleamos **d.mon** en el formato texto como en el gráfico en display monitors después tenemos varias opciones como son start, stop o select ya veremos que es de gran utilidad, de momento podemos lanzar uno de los 7 monitores que tenemos a nuestra disposición, para el entorno gráfico podemos introducir d.mon

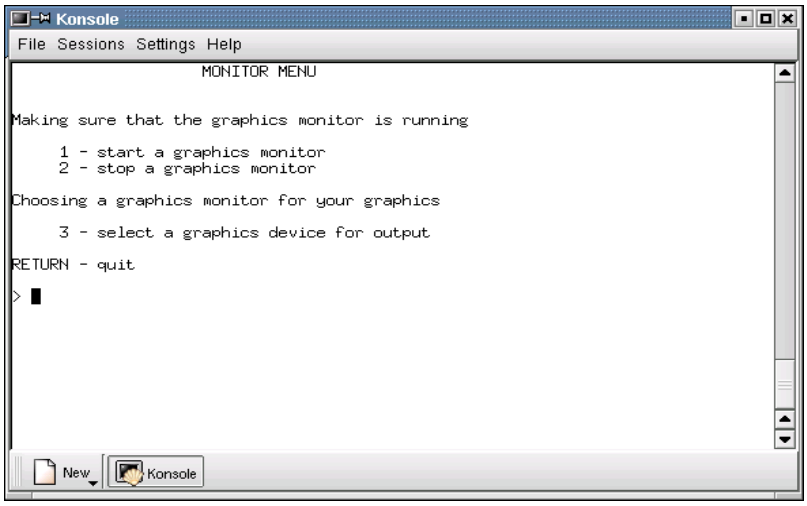


fig 2.4. menú del comando d.mon

permitiendo tener la posibilidad de lanzar un monitor pararlo o bien seleccionarlo, ¿cuál es la utilidad de estos últimos?; primero la facilidad de trabajar con hasta 7 monitores al mismo tiempo, los cuales iremos seleccionando para indicar al programas en cual de ellos queremos que el mapa se dibuje, la función stop nos va a permitir cerrar de forma adecuada los monitores que no estemos utilizando o a la hora de terminar la sesión cerrar estos, es importante cerrar correctamente GRASS ya que en el entorno de multiusuario tendria problemas otro usuario al iniciar una sesión de trabajo en la misma maquina.

Ahora vamos a practicar un poquito este comando, vamos a abrir un monitor **d.mon** y posteriormente opcion 1 y luego x3 por ejemplo o mas directamente podríamos teclear **d.mon** x3 y obtendríamos el mismo resultado. En este mismo sentido ocurre con la combinación de la orden para seleccionar y parar un monitor puede hacerse de forma rápida con la sintaxis **d.mon select=x3** o **d.mon stop=x3**. Es el objetivo pero por diferente camino lo mismo ocurre con el entorno gráfico, pero mas sencillo.

- **d.rast**, permite ver un mapa que exista en formato raster, que podríamos definirlo como el sistema de representación que divide el área de estudio en una agrupación de celdas cuadradas ordenadas en una secuencia especifica, cada una de estas celdas posee un solo valor que se considera representativo para el área que cubre esa celda. Este comando nos va a permitir ya empezar a ver cosas en grass y para ello solo hay que llamar al comando **d.rast** en la opción modo texto apareciendo una opción que nos permite introducir el nombre del comando o incluso hacer un listado previo para ver los mapas que tenemos en nuestro

MAPSET en formato raster, para los amantes del entorno gráfico simplemente tienen que presionar en la opción raster y posteriormente en display raster map, podemos ver dos cuadraditos uno en color normal que pone raster que es el que nos permite que salga la lista de los mapas y posteriormente siempre después de la elección correremos el comando pinchando en run(color azul).

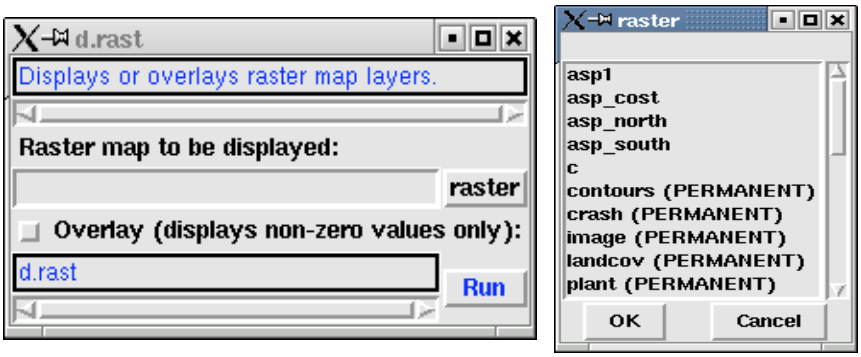


fig 2.5. Pantalla comando d.rast en modo gráfico

bueno vamos a verlo con un ejemplito, primero debemos lanzar un monitor **d.mon x3** por ejemplo, y después **d.rast** y veremos en el listado que en el caso del MAPSET seleccionado inicialmente que fue leics tenemos una serie de mapas pues bien vamos a ver alguno landcov, salen muchas opciones que de momento no vamos a tratar siempre le damos a la por defecto, también lo podemos hacer mas rápido con **d.rast landcov**.

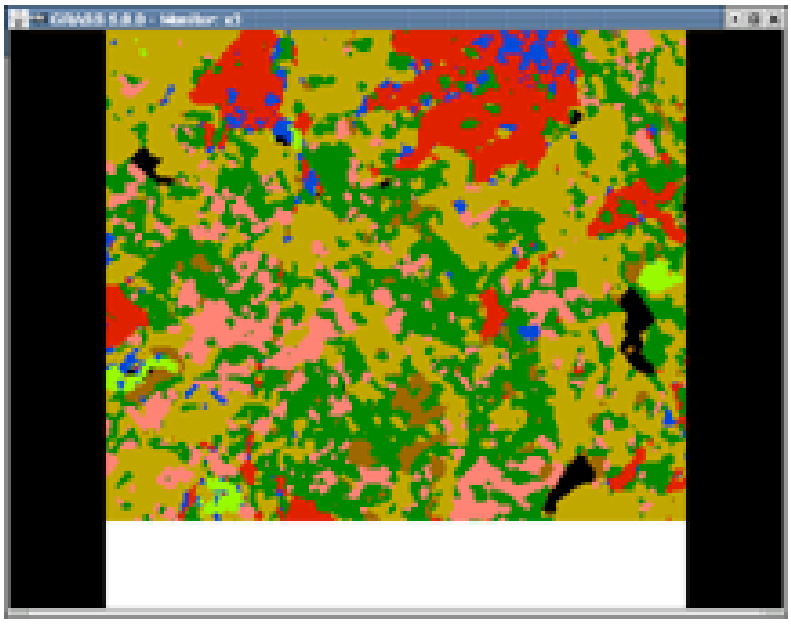


Fig 2.6. muestra en un monitor de un mapa en formato raster

dentro del comando **d.rast** vamos a ver una de sus opciones que es la opción -o que nos permite combinar mapas muy útil a la hora de superponerlos, así podemos ahora con esta opción superponer a landcov el mapa roads, ponemos **d.rast -o roads** y veremos un mapa superpuesto sobre el otro, ojo esto no es lo mismo que hacer **d.rast** primero con un mapa y luego con el otro, esto ultimo borraría el mapa anterior. Esta opción de superponer los mapas podemos hacerla permanente, es decir que quede guardado en un nuevo mapa el resultado con la orden **r.patch** así podemos indicar los mapas a combinar separados por comas(input) y el mapa de salida (output), **r.patch input=landcov,roads output=suelo_carretera**.

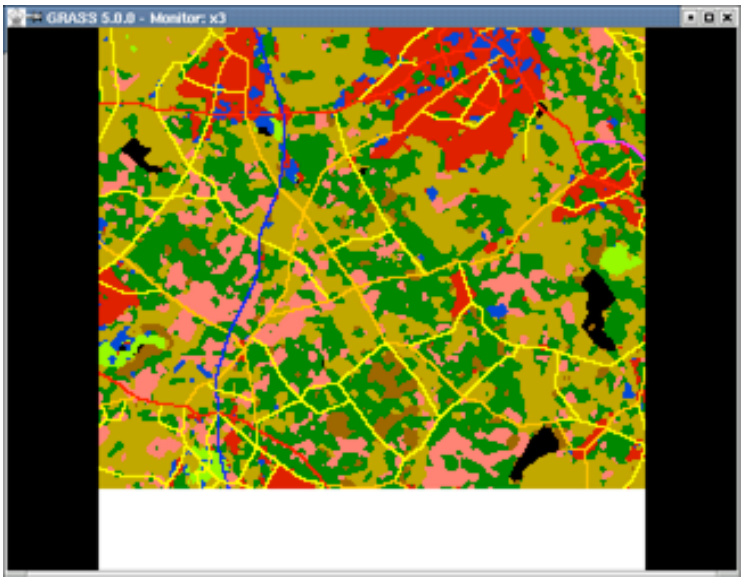


fig 2.7. superposición de dos mapas con el comando d.rast opción -o

- **d.vect**, permite al igual que el anterior comando ver un mapa en nuestro monitor pero en esta ocasión en formato vectorial, que se utiliza para representar, al contrario que el formato raster, responde a una concepción del mundo basada en objetos con limites definidos(lineas, polígonos,...) regidos por un modulo y una dirección. Al igual que en el caso anterior vamos directamente a ver como vemos un mapa, así en el entorno gráfico al igual que con raster vamos a la opción vector y seguimos los mismos pasos, igual ocurre con el modo texto, vemos la sintaxis **d.vect** en este caso nuestro MAPSET no posee ningún mapa en formato vectorial, bueno vamos a salir de GRASS ¡ojo cierra el monitor! **d.mon stop=x3** y luego exit. Reiniciamos GRASS con la LOCATION spearfish, que es otra LOCATION de ejemplo de GRASS, lanzamos un monitor y luego vemos **d.vect** y veremos que ya tenemos para elegir, así que **d.vect roads** que bien ya podemos ver tanto mapas en formato raster como en vectorial, ahora vamos a ver los dos modelos de datos a la vez, vamos a lanzar en

nuestro monitor **d.rast aspect** y luego directamente **d.vect roads** y veremos un mapa raster y encima superpuesto un vectorial que como es de lineas no oculta al mapa raster, pero ahora vamos a verlo mas bonito en vez de poner **d.vect roads**, vamos a poner **d.vect** y después roads para que salga todas esas preguntas que antes no hacíamos caso una de ellas nos permite elegir el color de las lineas del mapa que por defecto están en blanco escribiremos red y veremos que tal ahora.

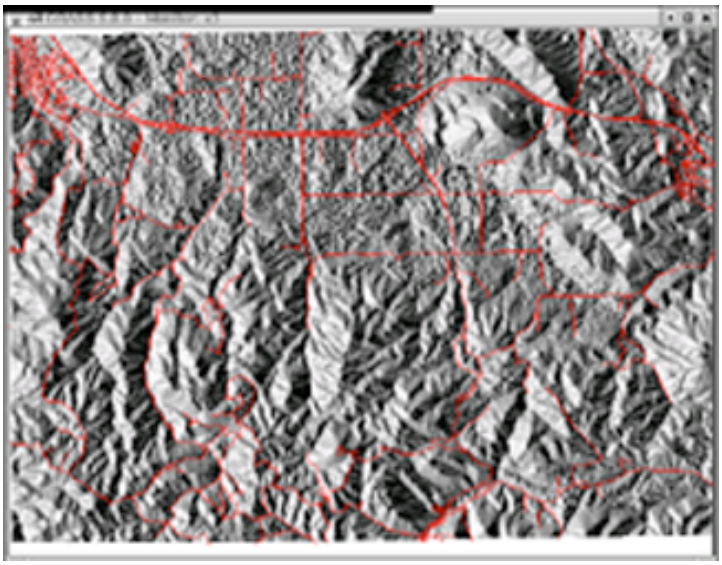


fig 2.8. aspecto de un mapa generado por combinación de un mapa vectorial y otro raster

podemos comprobar que si lo hacemos al revés el mapa raster taparía al mapa vectorial. Si queremos borrar el monitor para que vuelva a aparecer sin nada solo hay que llamar al comando **d.erase** en el caso del entorno gráfico este comando se puede localizar en la opción display en erase display frame.

- **d.sites** este comando nos va a permitir ver sites, puntos y localizaciones como son por ejemplo puntos tomados con un gps. Es la misma filosofía que los anteriores comandos así que vamos con un ejemplito **d.sites** luego hacemos un list para ver cuales son los mapas con sites que hay en nuestro MAPSET y veremos archsites, vemos que nos dan varias opciones como el color de los puntos y la representación en pantalla de estos.
- **d.erase** sirve para borrar el monitor seleccionado.

Bueno ya podemos salir de GRASS para poder comenzar con otro capitulo

Ahora vamos a hacer un pequeño ejercicio con los comandos aprendidos.

1.Iniciar GRASS en la LOCATION spearfish

2.Lanzar el monitor x3 y representar en el la imagen elevation.dem

3.Conocer la distribucion de las carreteras con respecto al mapa de pendientes superponiendo ambas imágenes.

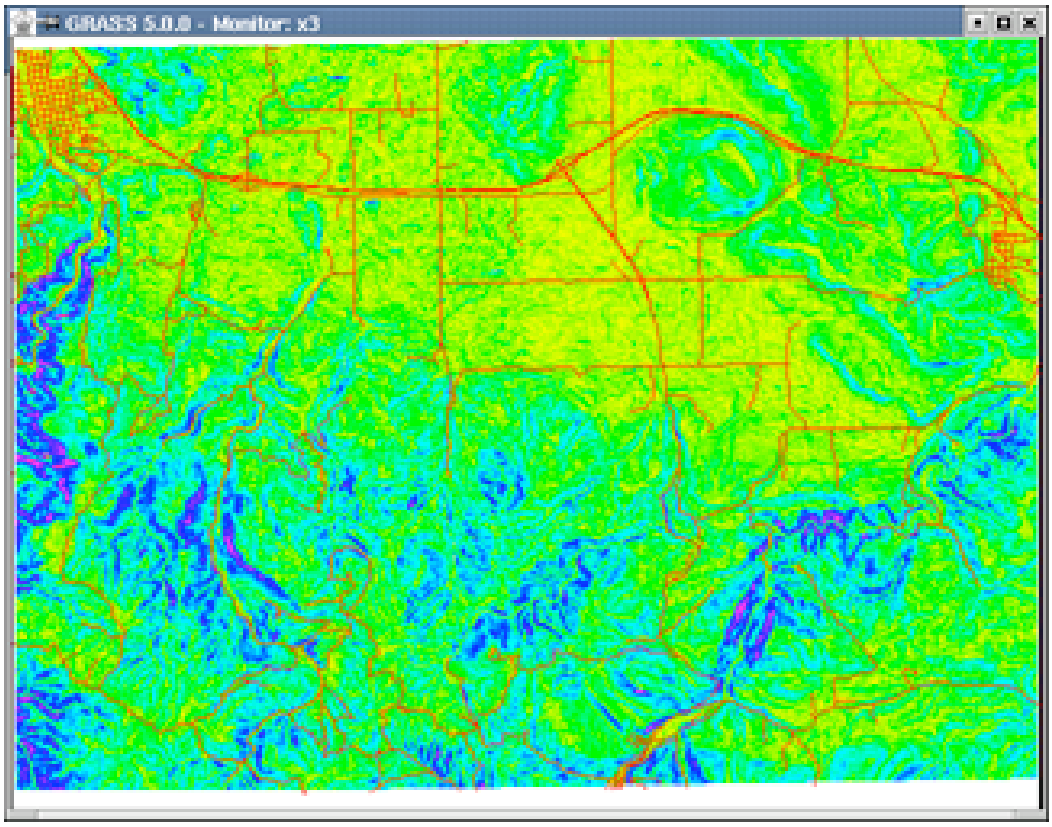


Fig. 2.9 Resultado del ejercicio superposición de mapas

Capitulo III: Consultas de mapas

Una vez que ya conocemos como visualizar mapas y hace algunas operaciones básicas de tratamiento de mapas, vamos a ver otro de los aspectos fundamentales de un sig como es la consulta de los valores en los mapas y la extracción de información de las capas que posee el mapa en cuestión, así en este capitulo vamos a tener los siguientes objetivos:

1. Obtener información de los mapas almacenados
2. Operar con los mapas, es decir, copiar, borrar y renombrar
3. Consultar los valores de las distintas celdillas de un mapa en formato raster y en formato vectorial.
4. Aprender el funcionamiento de los comandos:
    - **r.info**: Nos da información sobre capas raster
    - **v.info**: Nos da información sobre capas vectoriales
    - **s.info**: Nos da información sobre capas de sites
    - **g.remove**: Borra mapas
    - **g.rename**: Renombra mapas
    - **g.copy**: Copia mapas
    - **d.where**: Muestra las coordenadas de un punto en un mapa
    - **d.what.rast**: Muestra los valores de las capas donde si sitúa un punto seleccionado
    - **d.what.vect**: Muestra los valores de las capas donde si sitúa un punto seleccionado en un mapa vectorial
    - **d.what.sites**: Muestra los valores de los sites
    - **d.histogram**: Muestra el histograma de un mapa raster en el monitor activo
    - **d.legend**: Muestra la leyenda del mapa visualizado
    - **r.report**: Obtiene una tabla resuman de los datos de un fichero raster.

1. Obtener información de los mapas almacenados

Para conocer la información que contienen los diferentes mapas almacenados en nuestro DATABASE podemos utilizar cualquiera de los comandos que posee GRASS para estas consultas estos comandos son **r.info**, **v.info** y **s.info**.

Cada uno de estos comandos sirven para un tipo de formato de creación de mapas, así según sean raster, vectoriales o de sites.

Para este ejemplo podemos ver la información de los mapas que hemos utilizado en el capitulo anterior; elevation.dem y roads.

GRASS:~> **r.info elevation.dem**

obteniendo la siguiente información:

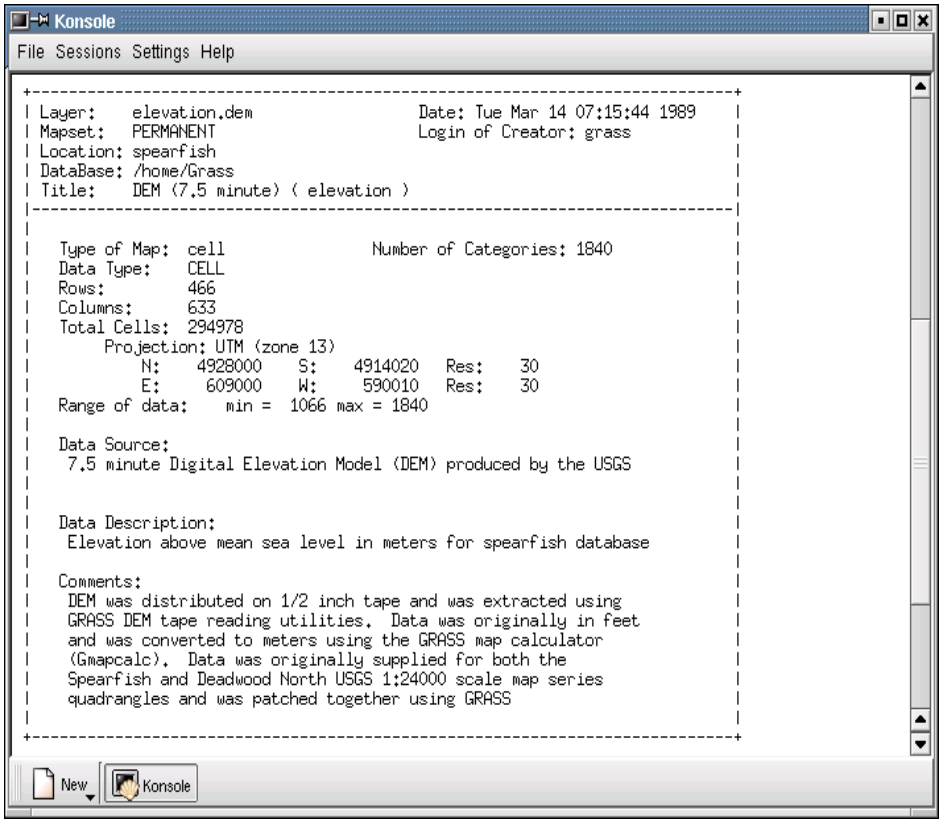


fig 3.1. Datos proporcionados por el comando **r.info**

en esta pantalla podemos ver mucha información sobre esta mapa, desde su LOCATION,MAPSET,... hasta el numero de columnas que tiene e incluso su región de trabajo sobre la cual hablaremos posteriormente.

Al igual que con el comando anterior podemos ver esta misma información de mapas vectoriales y sites para ello vamos a comprobar la información que nos dan los mapas roads y archsites

GRASS:~> **v.info roads**

GRASS:~> **s.info archsites**

2. Operar con los mapas, es decir, copiar, borrar y renombrar

Cada mapa de GRASS están compuestos por uno o varios ficheros que guardan distinta información fundamental para el funcionamiento de estos mapas, para ello vamos a ver un poco de la estructura de los ficheros de GRASS antes de entrar en los comandos de movimiento, borrado o copiado de los mapas.

Los modelos lógicos que utiliza GRASS como ya sabemos es el raster, vectorial(arco nodo) y sites (lista de puntos), en la siguiente figura podemos ver una muestra del árbol de fichero de GRASS

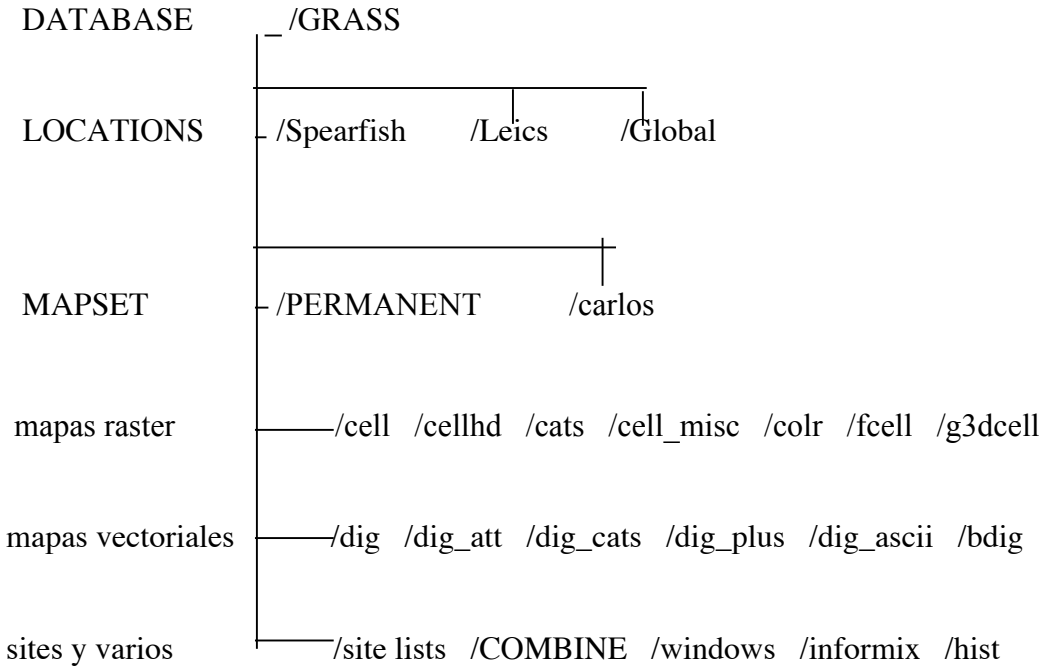


fig 3.2. Árbol de directorios de GRASS

estos datos están perfectamente organizados en carpetas que muestran diferentes informaciones así los directorios para el control de los mapas raster son:

- /cell que contiene los datos de los mapas raster en formato binario.
- /cellhd contiene la información para poder leer y ubicar espacialmente los datos contenidos en /cell
- /cats se guardan los significados, es decir etiquetas de los números almacenados en el directorio /cell
- /colr guarda información de la paleta de colores utilizada en cada mapa
- /cell_misc, /fcell y /g3dcell guardan informaciones mas concretas como visualizaciones que no vamos a ver por su complejidad

los directorios de control de mapas vectoriales son:

- /dig contiene los datos vectoriales en formato binario, es decir, valores de coordenadas que define objetos puntuales, lineales o poligonales.
- /dig_att contiene la información para poder leer y ubicar espacialmente los datos contenidos en /dig
- /dig_cats se almacenan datos similares a los que se guardaban en /cats, es decir, se hacen corresponder categorías numéricas o alfanuméricas a los identificadores definidos anteriormente.
- /dig_plus se almacenan ficheros que tienen un versión compactada, en un solo fichero, de la información de la información definida en cada uno de los ficheros anteriores. Se generan con un comando que veremos mas tarde que es **v.support** y es necesaria para cualquier consulta o análisis de datos esta en formato binario.
- /dig_ascii se guardan copias de los mapas que están en /dig pero en formato ascii, nos va a facilitar la exportación e importación de ficheros.

Por ultimo existen una serie de directorios para diferentes usos como combinación de mapas y por supuesto el directorio de información de las sites.

Con lo visto sobre los directorios que posee GRASS podemos decir que al hacer alguna de las operaciones sobre los mapas estaran involucrados una serie de ficheros en diferentes directorios. Con los comandos **g.copy, g.rename y g.remove** podemos hacer diferentes operaciones con tipos de ficheros como rast (raster), vect (vectoriales), icon (iconos), labels (def. Etiquetas), sites (puntos), región (def. Regiones), group (imágenes satélite) y 3dview (visualizaciones 3d).

Su funcionamiento es fácil y para verlo mejor podemos poner un ejemplo:

GRASS:~> g.copy rast=elevation.dem,elevación

con este mandato hemos copiado un mapa que se encuentra en el MAPSET PERMANNENT a mi MAPSET personal con un nuevo nombre que es elevación, fácil, así podemos luego manipular mejor los mapas ya son nuestros, por lo menos la copia, al igual ocurre con los comandos g.rename(renombra) y g.remove(borra), para los que están mas cómodos en el modo gráfico podemos indicar que estos comandos se encuentran en la barra principal en MAP.

3. Consultar los valores de las distintas celdillas de un mapa en formato raster y en formato vectorial.

Una vez visualizados los mapas resulta interesante saber algo sobre el tipo de información que contienen, para ello GRASS posee una serie de comandos como son:

- d.where este comando nos indica las coordenadas de los diferentes puntos donde pinchemos dentro del monitor gráfico en el que estemos viendo el mapa en cuestión, al ejecutar este comando nos aparece un menú donde nos explican en ingles el funcionamiento de este comando y los usos de los botones del ratón(recordemos que la opción tres botones puede no estar configurada en nuestro Linux esto lo podemos resolver pulsando los dos botones al mismo tiempo).

GRASS:~> **d.where elevation.dem**

Fig 3.3. Menú principal del comando d.where

como podemos observar podemos ver las coordenadas del punto seleccionado en el mapa. Existen otras ordenes mas completas que nos van a permitir conocer mas información sobre los mapas, estas son:

- d.what.rast, d.what.vect, d.what.sites que permiten obtener información sobre mapas raster, vectoriales y de sites respectivamente, aparecerá un menú de ratón idéntico al de d.where para ello vamos a ver un ejemplo

GRASS:~> d.what.vect roads

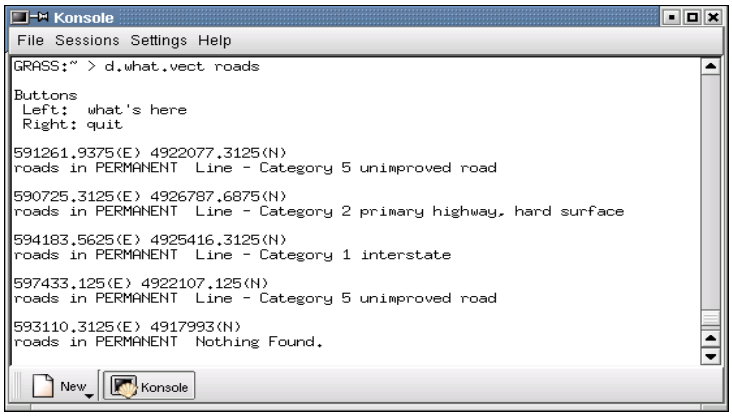


Fig 3.4. Resultado del comando d.what.vect para el mapa roads

para salir de estos comandos y volver a la terminal de texto debemos pulsar son el botón derecho sobre el mapa. Para utilizar estos comandos en menú gráfico los podemos utilizar por ejemplo en RASTER de la barra inicial luego analyse map y después en query whit mouse, idénticamente con VECTOR y SITE del menú principal.

También como comando de análisis de datos tenemos **d.measure** que nos permite medir longitudes y áreas de objetos que digitalizaremos nosotros sobre el monitor gráfico en este caso ira indicando cada una de las longitudes hasta terminar una linea o un polígono, el botón izquierdo nos dirá las coordenadas del punto en que pinchamos, con el derecho salimos del programa y el central determina el primer vértice y entra en un nuevo menú, con el botón central iremos definiendo una linea y/o un polígono interior a la misma. Cada nuevo vértice significa la adición de un nuevo tramo y el argumento de la longitud de la linea. Cuando finalmente pinchamos con el botón derecho, aunque no cerremos el polígono, aparecerán la longitud y el área del polígono en diferentes unidades junto a la posibilidad de volver a definir otro polígono o salir del programa.

Para finalizar este capitulo de modificación de datos de mapas veremos tres comandos bastante útiles como son:

- **d.histogram** este nos presenta en pantalla un histograma de frecuencia de los valores almacenados en el mapa al que hacemos referencia. Dibujara las barras con los mismos colores que aparecen en la imagen. Podemos ver un ejemplo del mapa elevation.dem, así tendríamos el mapa y su correspondiente histograma.(para poder ver ambas imágenes juntas debemos arrancar otro monitor, grass siempre escribirá sobre el ultimo mapa arrancado, para ir seleccionando los distintos monitores teniamos la herramienta d.mon select=x + numero de monitor

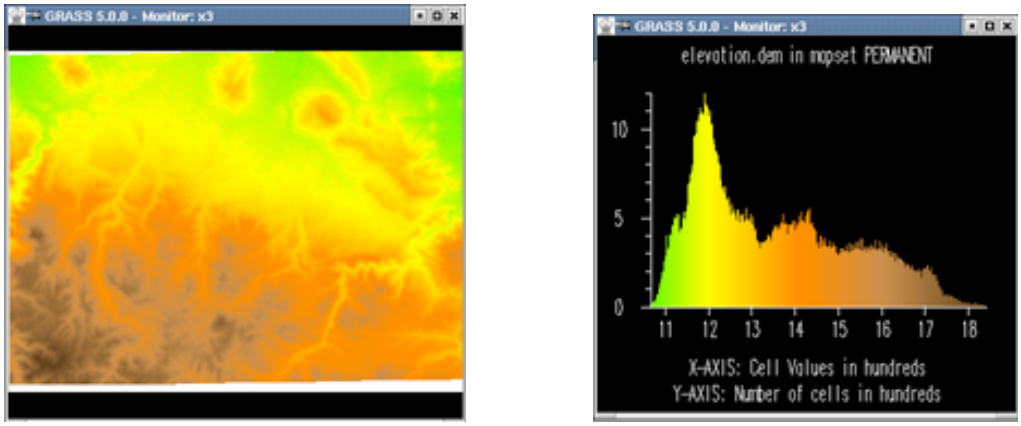


Fig 3.5. Mapa elevation.dem y su correspondiente histograma

aquí podemos ver el numero de celdas que existen de cada color, lo normal es que utilicemos también el comando **d.legend**, este nos permite conocer la leyenda del mapa que estamos viendo así podremos saber que nivel de elevación es la que mas abunda en la zona. Para ello utilizamos el comando con la sintaxis **d.legend elevation.dem**, ojo es bueno que borremos el mapa de nuestro monitor gráfico o tal vez mas útil seria lanzar otro monitor para tener ambas visiones juntas.

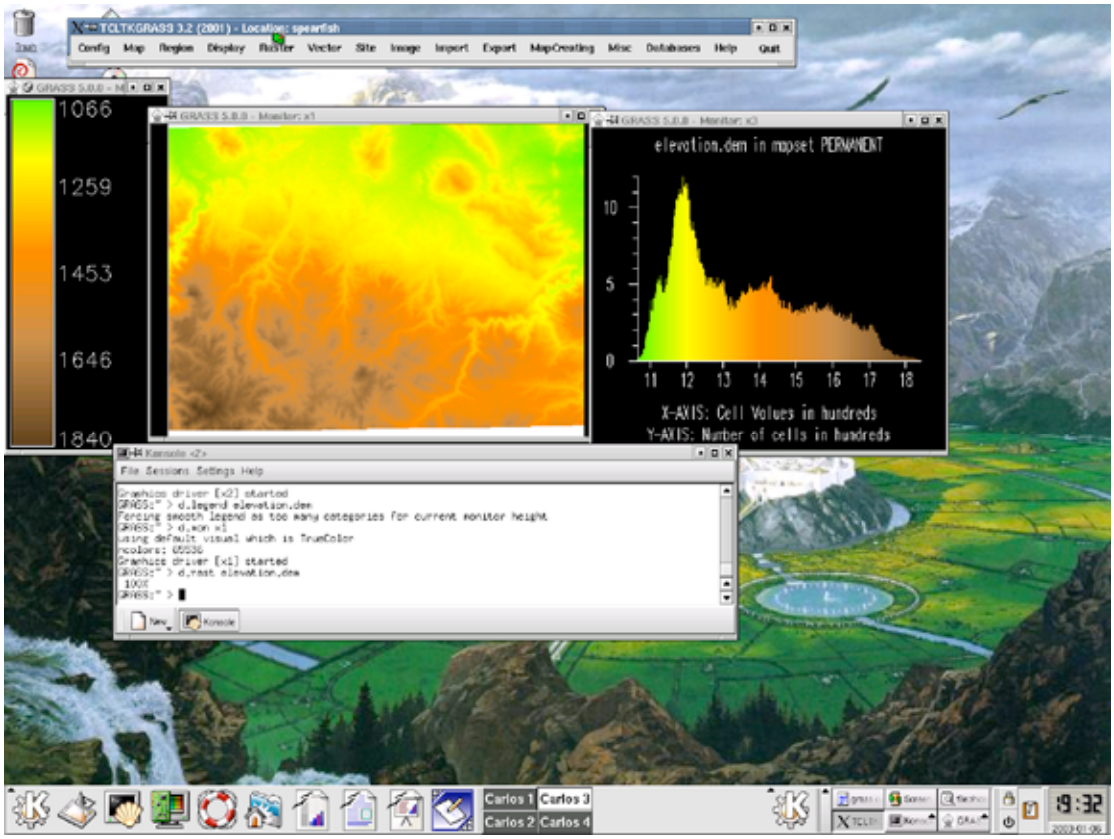


Fig 3.6. Funcionamiento de GRASS con multipantallas

Puede ser difícil de comprender pero el hecho de que GRASS permita trabajar, según las versiones, con numerosos monitores gráficos es una ventaja que no debemos olvidar lo único que debemos tener en cuenta es la necesidad de seleccionar el monitor en el que vamos a lanzar la imagen ya que si no lo hacemos nos la mostrara en el ultimo monitor abierto. Ambos comandos los podemos ver en el entorno gráfico en DISPLAYS y graphics.

Por ultimo veremos el comando **r.report**, que permite obtener el área ocupada por las diferentes combinaciones de categorías, pueden seleccionarse diferentes unidades para medir áreas, vamos a ver con esto las diferentes áreas del mapa de elevation.dem, principalmente la sintaxis es **r.report elevation.dem units**=unidades tales como mi(millas cuadradas), m(metros), K(kilómetros), a(acres), ha(hectares), c(celdillas) y p(porcentaje), pueden

introducirse varias medidas solo basta separarlas con comas, en esta caso queremos ver los valores del mapa elevation.dem en kilómetros y en porcentaje.

GRASS:~> **r.report elevation.dem units=k,p**

hemos seleccionado un mapa con muchas celdas y casi ni hemos visto las celdas si tenemos ese problemas podemos bien jugar con la barra de nuestra terminal o poner al final del comando la opción de Linux more.

|

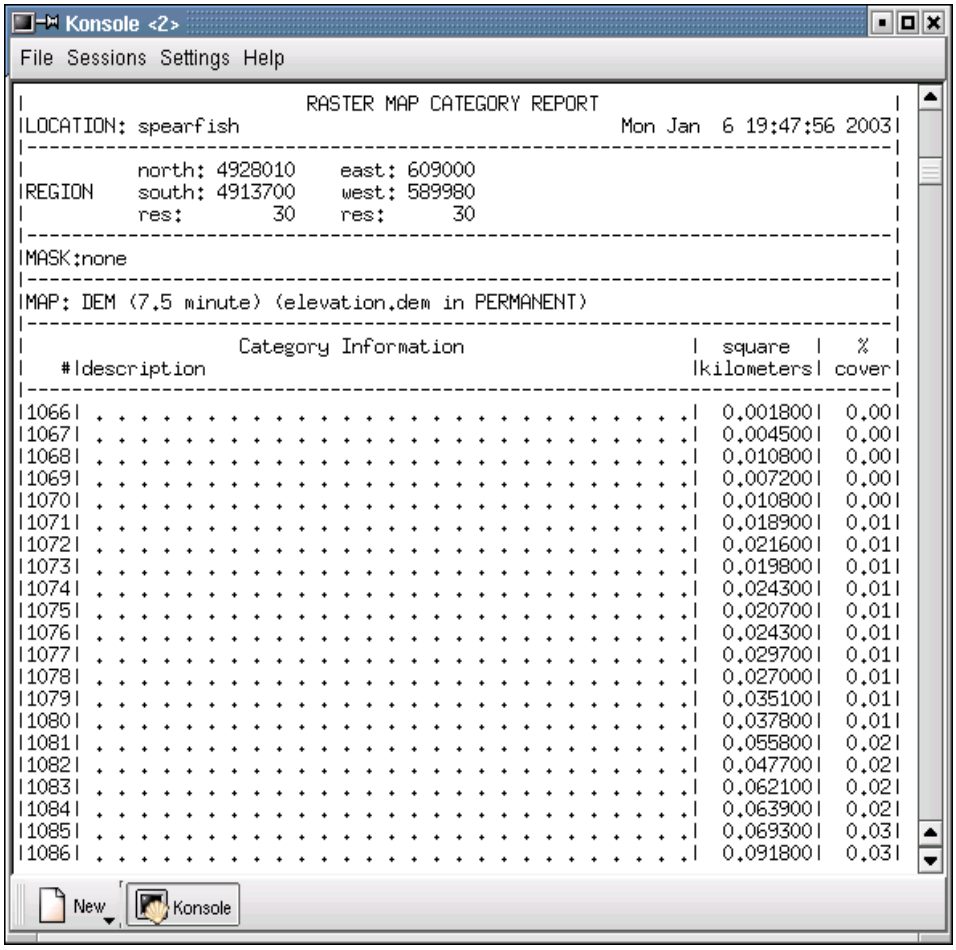


Fig 3.7. Resultado del análisis del mapa elevation.dem en kilómetros y porcentaje de celdas ocupadas por cada valor.

Ahora vamos a hacer un pequeño ejercicio con los comandos aprendidos.

1. Indicar la región en que se sitúa el mapa de usos de suelos (landuse)
2. Copiar el mapa landuse y renombrarlo dentro de nuestro MAPSET como usos
3. Determinar el porcentaje de zonas residenciales de la zona y su superficie en hectareas
4. Medir la linea de transporte industrial sobre el mapa landuse

Capitulo IV: Visualización de mapas

Esta capitulo principalmente tratara de mostrarnos las herramientas de visualización de la información espacial. Como objetivos podemos distinguir:

1. Manejar leyendas y paletas de colores
2. Modificar el área de trabajo
3. Hacer vistas en 3D
4. Manejar diferentes comandos de GRASS como son:
    - **d.zoom**: Permite definir una ventana para ver mas cerca una zona del mapa en cuestión
    - **d.legend**: Obtiene la leyenda de un mapa en un monitor gráfico
    - **d.3d**: Representación de un mapa en 3D
    - **g.region**: Modifica la región de trabajo
    - **r.mask**: Crea una mascara para limitar la visualización y cualquier operación de álgebra a un área irregular
    - **r.colors**: Asigna una paleta de color a un mapa
    - **d.rast.num**: Permite conocer los valores de los pixels asociados a cada celda de un mapa

1. Manejar leyendas y paletas de colores

En el anterior capítulo ya comentamos las grandes posibilidades que tiene GRASS para trabajar con múltiples monitores a la vez, la cual deberíamos acostumbrarnos a utilizar, también sabemos que los monitores no tienen un dimensión fija podemos tanto moverlos como agrandarlos con un simple movimiento de ratón. Uno de los comandos que también vimos en el anterior capitulo fue el referente a la leyenda esta se visualiza con el comando **d.legend**, perro al lanzar este comando sin argumentos podemos encontrarnos con un problema generado por un numero de categorías elevado o muy pequeño para el tamaño de la pantalla. Para solucionar este problema podemos utilizar el comando **d.legend** con la opción lines=numero; donde numero las categorías que deberían entrar en la pantalla. Si el numero de categorías reales es mayor que el numero, entonces solo numero de categorías serán mostradas.

2. Modificar el área de trabajo

Normalmente las capas que se utilizan en SIG suelen ser muy grandes y deben visualizarse solo cierto numero de pixels. Para poder ver en detalle alguna zona en particular del mapa podemos utilizar el comando **d.zoom**, es un comando interactivo fácil de usar con el ratón y que permite ampliar una zona que marquemos, iniciando el primer punto del rectángulo con el botón izquierdo y señalando el segundo punto con el botón del medio. Después de ejecutar la opción **d.zoom** debe borrarse la pantalla para volver a dibujar el mapa en el monitor gráfico. Debemos saber que este zoom permanecerá constante en todo momento con los mapas que visualicemos, mediante el comando **g.region -d**, pero ahora veremos este comando en mayor profundidad. Tambien dentro de zoom tenemos la posibilidad de hacer zoom out y ver más lejos la zona del mapa, pulsando el boton central al cargar el comendo zoom.

Una de las grandes ventajas que posee GRASS es la de poder superponer distintos mapas con diferentes características geométricas, es decir, diferentes limites y diferente resolución)

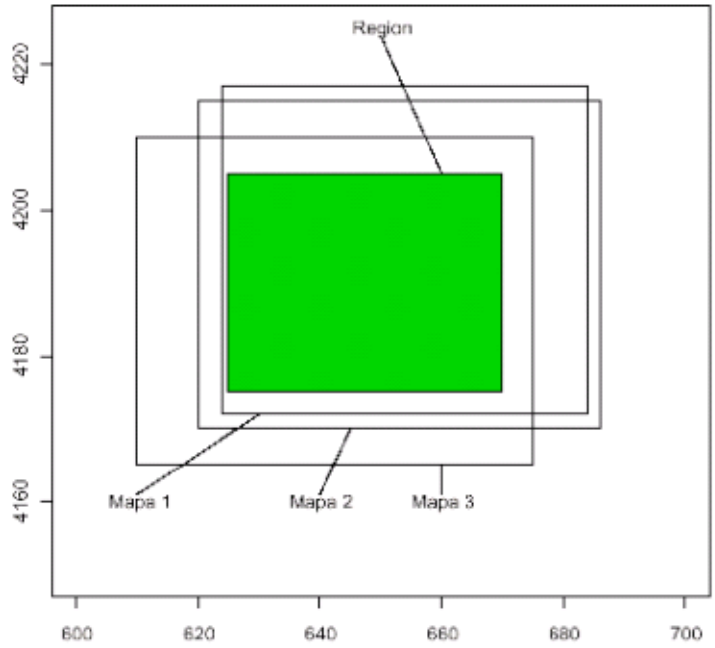


fig. 4.1. Superposición de mapas y región con diferentes características

Por tanto, las características geométricas de la pantalla de visualización deben ser independientes de la de los mapas visualizados y se conoce como región. La región puede modificarse mediante la orden **d.zoom** o bien con la orden **g.region** (debemos tener en cuenta que las modificaciones en la región no solo afectan a la visualización sino a todas las operaciones que se desarrollen tras su modificación, por tanto hay que andar con pies de plomo para su manejo.

El comando g.region puede utilizarse con sintaxis muy abundante con opciones como:

* -d que vuelve a la región por defecto

* -p Devuelve las características de la región actual

* -g Devuelve las características de la región actual(en modo script)

* -u No actualiza la región.

Y también contiene una serie de parámetros como son:

•región: Indica un fichero de región del que tomar los parámetros

•raster: Copia la región de un mapa raster

•vector: Copia la región de un mapa vectorial

•sites: Copia la región de un mapa de sitios

•3dview: Optimiza la región para una vista 3D

•n: Valor del limite Norte

•s: Valor del limite Sur

•e: Valor del limite Este

•w: Valor del limite Oeste

•res: Resolución

•nsres: Resolución norte-sur

•ewres: Resolución este-oeste

•zoom

•align: Alinea mapas raster

•save: Graba la región actual a un fichero

Tal vez esta forma de actuar con el comando g.region nos sea útil para cuando dominemos el programa en este momento lo mas útil sera utilizar el menú contextual que nos sale por defecto al dar la opción g.region sin argumentos

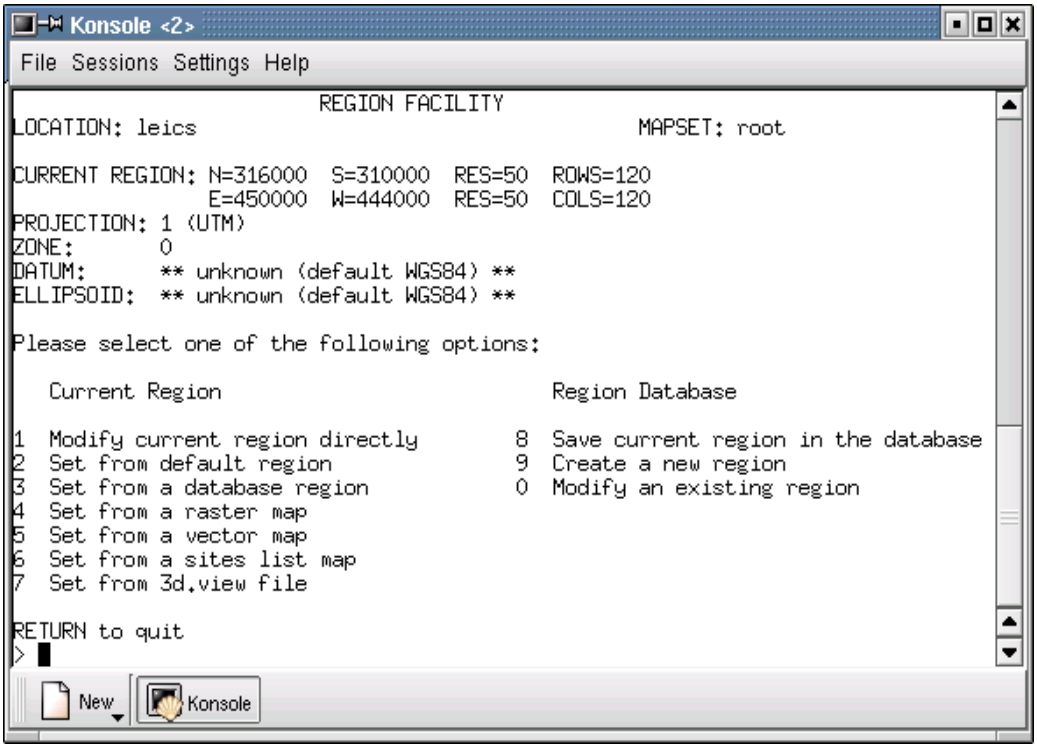


fig. 4.2. Menú para la modificación de la región de trabajo con g.region

Si elegimos la opción de modificar la región actual directamente veremos:

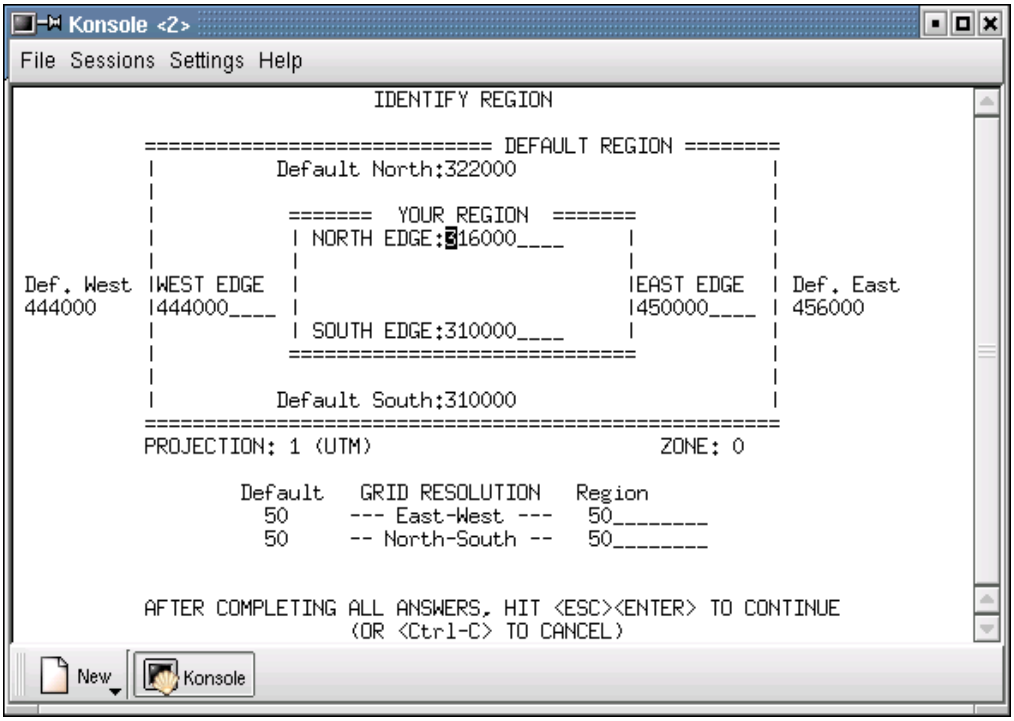


fig. 4.3. Modificando la región de trabajo

Podemos tal vez entender mejor lo que significa esto. Podemos ver que existe una región por defecto(que seria si ponemos un ejemplo nuestra casa) y una región propia(que seria en el ejemplo nuestra habitación) va a ser la región en donde trabajaremos, con lo que es obvio que los limites de nuestra región deben estar dentro de la región por defecto, con lo que podemos manejar los limites Norte, Sur, Este y Oeste de nuestra región y también manipular la resolución de las celdas. Posteriormente podremos ver las utilidades que nos muestra este comando, ojo que al igual que hablamos al principio en el capitulo de introducción esto es modo texto y tendremos que tener en cuenta todos los avisos que dimos en el momento que explicamos como introducir el MAPSET, LOCATION, etc. Para volver al menú principal solo hay que pulsar **<esc>** y **<enter>.**

De este modo el manejo adecuado de la región activa nos permite centrar el trabajo en áreas concretas e incluso con resoluciones concretas ya que podemos alterar a voluntad el tamaño de los pixels. En todo caso esta ultima posibilidad debe utilizarse razonablemente. No tendría sentido trabajar con tamaños de pixel de 10 metros si nuestros datos originales tienen una resolución de 100 metros.

Este comando nos va a permitir poder explicar otro comando que nos va a hacer comprender mejor lo que es un mapa raster, este comando es **d.rast.num** que nos va a mostrar en pantalla los números de los pixel de los mapas que estamos viendo, en este caso como podemos ver en el ejercicio final del capitulo.

fig. 4.4. Región de trabajo después de realizar un zoom

Este gráfico surge de realizar un zoom en el mapa landuse en la zona residencial(situará en el norte) y posteriormente analizados los valores de los pixels.

También ahora que conocemos el concepto de región, podemos introducir el concepto de mascaras. Con los comandos **d.zoom** y **g.region** solo podemos trabajar con áreas rectangulares y en muchos casos lo mas útil seria trabajar con áreas irregulares con algunas características en común(capas). Para ello se hace necesario definir una mascara que indique los lugares en los que GRASS debe centrar tanto la visualización como los análisis posteriores. Para conseguir este objetivo podemos utilizar el comando r.mask que nos permite trabajar solo con las capas que queramos de un mapa, es decir en un mapa como el de landuse queremos trabajar solo con las zonas urbanas y con las lineas de transporte, podemos seleccionar unas capas determinadas con la opción identify a new mask seleccionando el mapa a realizar la mascara, todas las operaciones realizadas a partir de ese momento se realicen sobre estas capas unicamente, esto terminará cuando eliminemos la mascara con r.mask y luego eligiendo la opción remove.

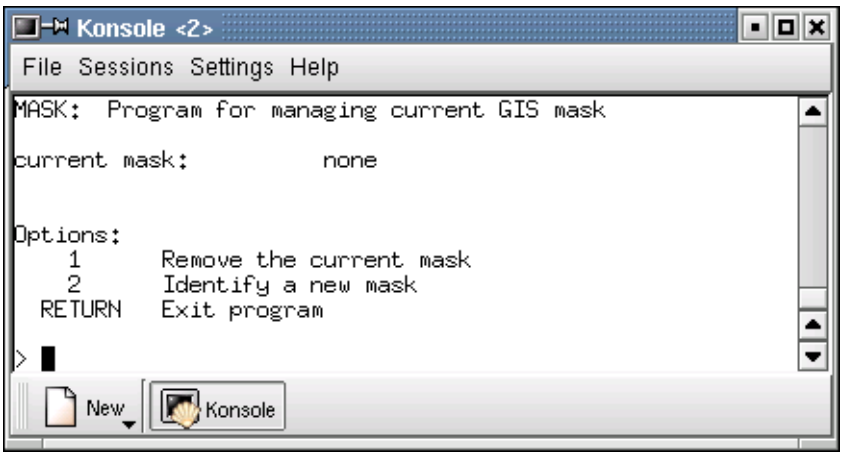


fig. 4.5. Menú interactivo de r.mask

Este menú nos indica cual es la mascara actual(current mask), si no existe ninguna definida nos indica la opción none. Luego presenta dos opciones, la primera desactiva la mascara actual, la segunda define una nueva mascara. Si seleccionamos la segunda opción debe elegirse el mapa que actuara como mascara y a continuación determinar que valores del mapa seleccionado se consideran incluidos en el área de trabajo y cuales no. En definitiva se construye una variable binomial de pertenencia o no al área de trabajo. Posteriormente veremos otros tipos de comandos que nos van a permitir crear áreas digitalizadas a partir de un mapa raster.

Los mapas raster deben tener una paleta de colores asignada que permite la conversión de los valores codificados en colores para su representación en un monitor. La mayor parte de los sistemas de vídeo, representan los colores mediante el sistema RGB, es decir, como combinación de los colores primarios verde, rojo y azul. A cada uno de estos colores se le asigna una intensidad, que en el caso de las tarjetas de vídeo actuales va desde 0 a 255, de la mezcla de estos tres colores con sus intensidades resultan 256^3. Estos valores se almacenan en un fichero especial de colores en el directorio colr del MAPSET donde esta el mapa. Para manejar las distintas paletas de colores que tenemos en los mapas raster podemos utilizar numerosos comandos dentro de GRASS5 como son:

- r.colors: permite asignar al mapa diferentes paletas temáticas entre las que podemos elegir están:

 * aspect: paleta utilizada principalmente para representar orientaciones
 * grey: paleta para niveles de gris
 * grey.eq: paleta que representa niveles de gris con histograma ecualizado
 * gyr: paleta verde-amarillo-rojo
 * rainbow: paleta de colores arco iris
 * ramp: paleta de colores en rampa
 * ryg: paleta rojo-amarillo-rojo
 * random: paleta aleatoria, útil para variables cualitativas con pocos niveles
 * wave: repite los mismos colores varias veces, útil para variables cuantitativas con un rango muy amplio
 * rules: crea los colores interactivos o mediante un fichero(opción que se escapa a este curso de iniciación).

- d.colorlist: podemos ver la lista de colores que tienen el mapa
- d.colors: cambia interactivamente los colores de un mapa en pantalla
- d.colormode: Permite establecer si un mapa podría ser expuesto usándolo con la propia tabla de color o la tabla de color fijo del monitor gráfico, con más de 16 colores se utiliza el modo de color float y a menos el modo de color fixed. Siempre que se efectue alguna operación de cambio de color en el mapa debemos volver a cargar el mapa en el monitor para ver los cambios.

3. Hacer vistas en 3D

Son una de las representaciones más espectaculares que pueden conseguirse con un SIG. En

GRASS se dispone de un modulo llamado d.3d y que permite realizar estas vistas a partir primero de un mapa que va a servir de como de manta de color encima del mapa de elevaciones. Una vez introducidos ambos parámetros, en modo texto aparece un menú donde iremos colocando los distintos parámetros de la vista.

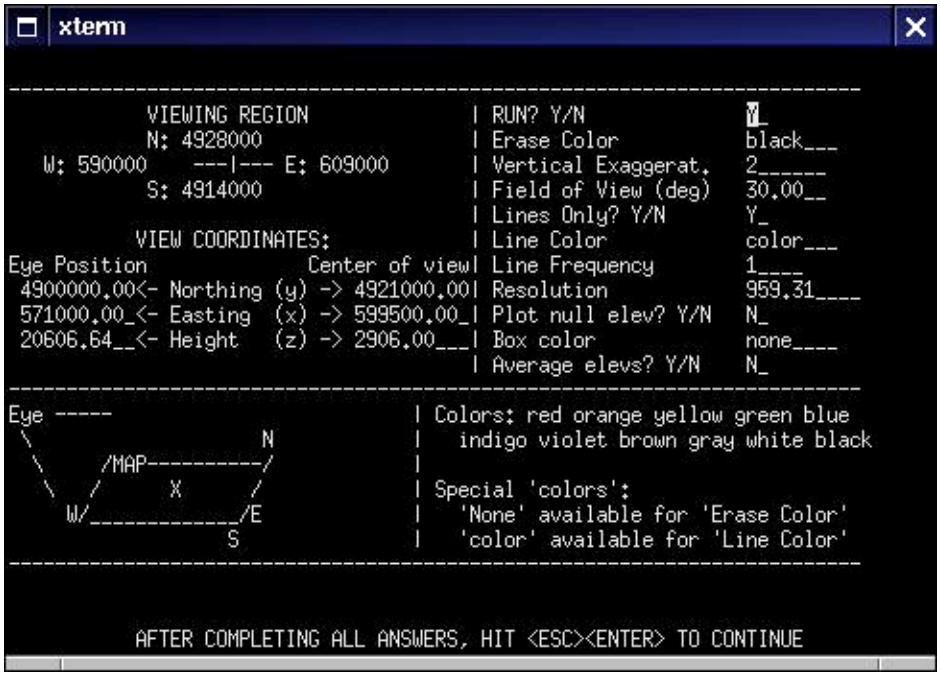


fig. 4.6. Menú interactivo de d.3d

Es un modo texto donde podemos ir indicando cosas como la exageración vertical, el punto de vista, si queremos que el mapa sea de lineas o de pixels coloreados, que resolución queremos y más parámetros que iremos descubriendo al hacer las diferentes vistas de nuestro mapa en 3D, solo debemos de tener en cuenta que para terminar de utilizar este comando deberemos sustituir en RUN Y por N. Es importante para este comando probar y jugar con el sin miedo. Para los más reticentes al entorno de comandos tenemos este en la barra de herramientas en Raster, display y display 3-d images, resultara mucho más sencillo de utilizar, pero nos permitirá tal vez jugar menos con las vistas.

Por ultimo solo queda presentar un comando que parte de una de las opciones de d.rast que es d.rast.num en este caso nos va a permitir entender mejor lo que es un mapa raster, opción que la dejaremos para los ejercicios.

Bueno ya podemos salir de GRASS para poder comenzar con otro capitulo

Ahora vamos a hacer un pequeño ejercicio con los comandos aprendidos.

Ejercicios:

1.Representar los mapas elevation.dem y geology con diferentes paletas de color, a fin de determinar cuales son más apropiadas para variables cuantitativas y cualitativas.

2.Utilizar los comandos para ver la leyenda del mapa elevation.dem y utilizar sus diferentes parámetros.

3.Con el mapa de landuse efectuar un zoom sobre la zona donde se sitúan más las zonas residenciales, posteriormente con ese zoom, vamos a conocer cuales son los números de cada pixel de la zona en cuestión(comando d.rast.num)

4.Observar el mapa de suelos(soils)en una vista tridimensional, utilizando como base de color el mapa de suelos y como el de elevaciones elevation.dem

fig. 4.7 resultado del ejercicio numero 3

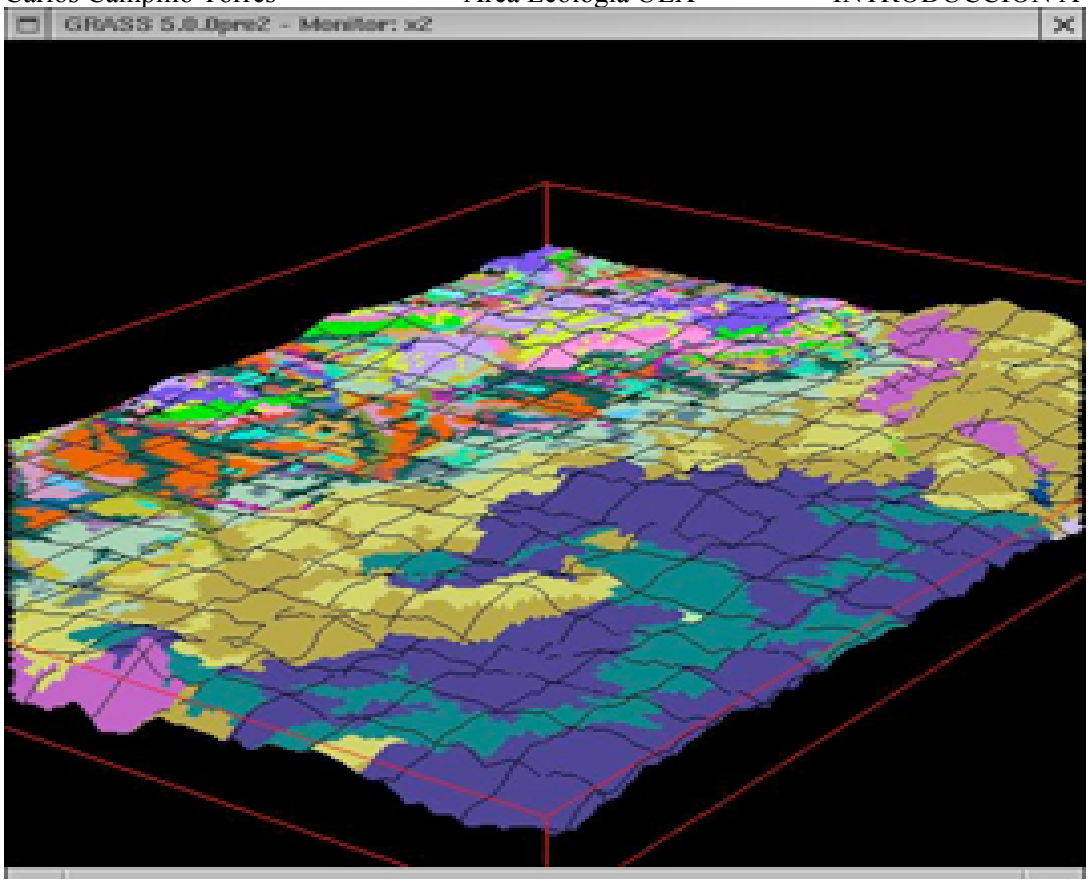


fig 4.8 Resultado ejercicio 4 vista 3D

Capitulo V: Operando con mapas

Esta capitulo principalmente tratara de mostrarnos las herramientas de álgebra de mapas como principales objetivos podemos distinguir:

1. Breves nociones sobre el concepto de operador y tipos
2. Manejar diferentes comandos de GRASS como son:
    - r.reclass: Permite reclasificar los valores de un mapa
    - r.cross: Crea un mapa en el que se identifica cada combinación individual de las categorías presentes en los mapas de entrada
    - r.neighbors: Da a cada celda un valor que es función de los valores de las celdas circundantes
    - r.mfilter: Permite crear operadores de vecindad que calculan una media ponderada de los valores de los pixels circundantes.
    - r.slope.aspect: Genera mapas de pendientes y orientaciones a partir de un Modelo Digital de Elevaciones.
    - r.clump: Reclasifica los datos en un mapa raster agrupando celdas contiguas que contienen el mismo valor.
    - r.buffer: Crea zonas buffer(áreas tampón) alrededor de un objeto.
    - r.average: Crea un mapa con los valores medios de una variable cuantitativa para distintos niveles de una variable cualitativa.
    - r.mapcalc: Lenguaje de programación del álgebra de mapas.

1. Breves nociones sobre el concepto de operador y tipos

Llegamos a un capitulo donde pasamos del simple análisis y visualización de los mapas y sus capas al completo análisis de mapas en GRASS.

Para superar la fase de presentación y consulta de datos en un SIG e iniciar un uso más avanzado, como es análisis, modelización de procesos, toma de decisiones, suma de mapas, etc, es necesaria la utilización del álgebra de mapas. Uno de los conceptos fundamentales en este álgebra de mapas es el de operador, entendido este como cada una de las operaciones que a partir de uno o más mapas de entrada y texto producen uno o más mapas de salida, o incluso utilizando la salida en formato texto

El desarrollo de un proyecto SIG consistiría de esta manera en la división sucesiva de un

problema en subtareas cada vez más simples hasta el momento en que cada una de estas pudiera expresarse como un operador de álgebra de mapas.

Sin duda el comando más potente de los que podemos utilizar para el álgebra de mapas en GRASS es r.mapcalc. Este es un interprete que permite ejecutar cualquier tipo de operador local o de vecindad. En GRASS también poseemos comandos para determinados análisis específicos (reclasificaciones, reescalado, filtrado, pendientes y orientaciones,...).

Ahora vamos a hablar sobre el concepto de operadores locales:

Los operadores locales son aquellos que asignan a cada celdilla del mapa un valor obtenido de los valores de la misma celdilla en otros mapas. El comando más sencillo es r.reclass (reclasifica mapas), el comando r.cross (cruza los mapas y crea uno nuevo con todas las posibilidades). Vamos a ver un poquito las sintaxis de cada uno de ellos:

- r.reclass: Permite la reclasificación de un mapa, permitiendo modificar una paleta de colores y así transformar un mapa de variables cuantitativas a uno de variables cualitativas con una leyenda más corta. Este comando es un operador local de álgebra de mapas que se fundamenta en que asigna a cada celdilla del nuevo mapa un valor en función del rango de valores al que pertenezca esa misma celdilla en el mapa de entrada. Vamos a ver un ejemplo para orientarnos más, tenemos el mapa de elevation.dem que tiene muchos valores de pendientes y lo vamos a reclasificar en tres categorías:

+ Zona menor 1000 metros
+ Zona Montañosa media : 1000-1400 metros
+ Zona Montañosa alta: >1400 metros

Para ello la sintaxis sería r.reclass input=elevation.dem output=elevación, con este comando saldría una pantalla que nos permitiría ir metiendo las reglas de la leyenda y colores del mapa, en este caso sería

> 0 thru 1000 = 1 Zona baja
> 1000 thru 1400 = 2 Montaña media
> 1400 thru 2000 = 3 Montaña alta
> end

fig 5.1 reclasificación elevaciones

Con esto conseguimos que un mapa que tenia tantos colores como números de altitud y que no permitía ver toda la leyenda, lo transformemos en un mapa más manejable, con solo tres categorías. El comando r.reclass puede ser usado también interactivamente, solo basta poner solamente r.reclass e ir siguiendo las indicaciones, en el caso de categorías muy grandes puede ser muy pesado, pero muy útil en mapas con pocas categorías.

- r.cross: Permite crear un mapa en el que se identifica cada combinación individual de las categorías presentes en los mapas de entrada. Si tenemos un mapa de usos del suelo y otro de litología podríamos extraer cualquier combinación de valores de litología y usos de suelo presente en los mapas originales, podemos combinar hasta 10 mapas raster diferentes.

Vamos a ver un ejemplo; probemos con combinar dos mapas como son el de suelos(soils) y el de carreteras (roads). Veremos un mapa que nos realiza un análisis de los dos mapas, es importante utilizar este comando para análisis y ver lo que va saliendo con la combinación de los distintos mapas de la zona.

Dentro de las posibilidades de operadores locales tenemos también los operadores de vecindad, como son los comandos:

- r.neighbors: Este comando da a cada celda un valor que esta en función de los valores de las celdas circundantes, este comando dentro de su sintaxis permite definir diferentes parámetros como son:

-a : no alinear salida con entrada

-q : modo silencioso

-z : preservar ceros en el mapa de salida (por defecto se transforman en nulos)

*input : mapa raster existente

*output : mapa raster que se va a crear

*method : permite realizar operaciones como: promedio(average), media(median), moda(mode), mínimo(minimum), máximo(maximum), derivada estandar(stddev), varianza(varianze), diversidad(diversity) e interspersion.

*size : radio de vecindad con varias opciones (1, 3, 5, 7, 9, 11, 13, 15, 17 ,19 ,21, 23, 25)

podemos probar este comando interactivo con la orden r.neighbors con el mapa roads veremos que después de utilizar el comando cambiara nuestro mapas con relación a que se pintaran trozos de carretera que no existían antes como podemos ver en el ejemplo.

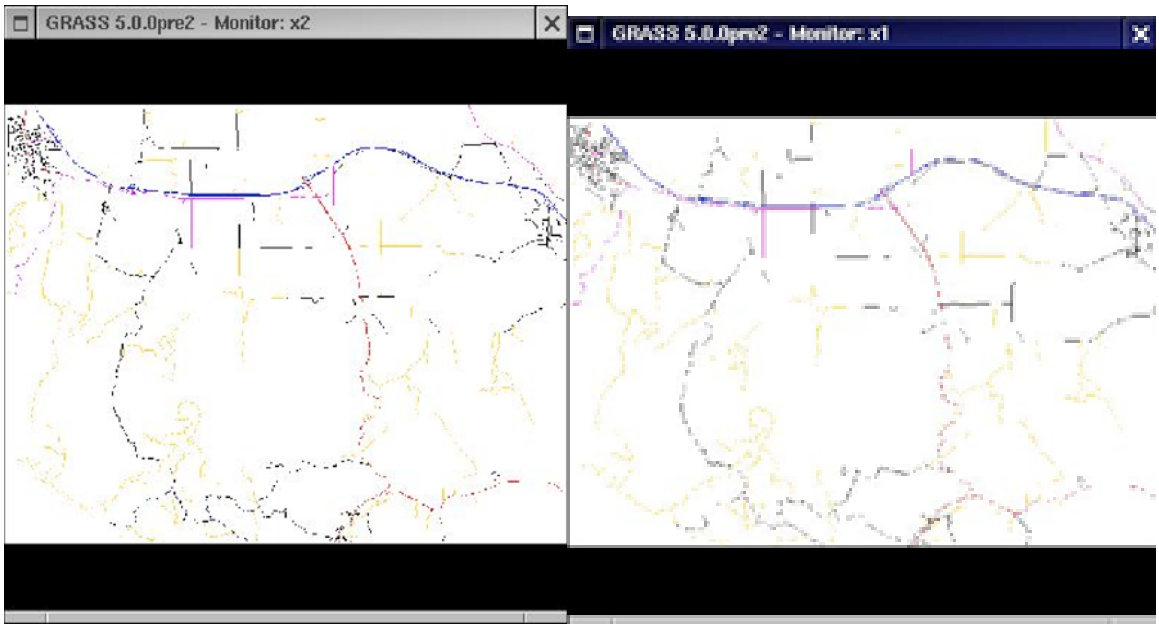


Fig 5.2 Aplicación del comando r.neighbors sobre el mapa raster roads

- r.mfilter : Permite crear operadores de vecindad que calculan una media ponderada de los valores de los pixels circundantes funciona creando una matriz en un archivo de texto,

- r.slope.aspect : Este comando genera mapas de pendientes y orientaciones a partir de un Modelo Digital de Elevaciones, acarrea una serie de parámetros que irán definiéndonos lo que hacer con ese MDE.

* elevation: mapa que contiene las elevaciones
* slope: mapa de pendientes
* format: formato del mapa de pendientes(opciones degrees y percent)
* prec: tipo de salida para, pendiente y orientaciones(opciones double,float e int)
* aspect: mapa de orientaciones
* pcurv: mapa de curvatura longitudinal a la pendiente
* tcurv: mapa de curvatura perpendicular a la pendiente
* dx: pendiente a la dirección E-W
* dy: pendiente en la dirección N-S
* dxx: derivada parcial de la pendiente en X
* dyy: derivada parcial de la pendiente en Y
* dxy: derivada parcial de la pendiente en XY
* zfactor: factor multiplicador para convertir las unidades de elevación en metros
* min_slp_allowed: valor mínimo que debe tener la pendiente para calcular la orientación.

- r.clump: permite reclasificar los datos de un mapa raster agrupando celdas contiguas que contienen el mismo valor dándoles un valor único. La utilidad principal de este comando es cuando necesitamos individualizar manchas homogéneas, si queremos analizar una parcela de matorral por separado en lugar de todo el suelo ocupado por matorral en conjunto.

- r.buffer : Crea zonas buffer(tampón) alrededor de un objeto. Se define como objeto en un mapa raster un conjunto de pixels adyacentes con valores nulo

Dentro de los distintos operadores también tenemos los de área que contienen diferentes comandos en GRASS como son:

- r.average: este comando permite obtener información de tipo estadística de un mapa raster, importante para la utilización de este comando es tener un mapa base con información de tipo cualitativa y un mapa con la información cuantitativa. Vamos a ver la información estadística que posee el mapa elevation.dem para diferentes usos del suelo(landuse).

Sintaxis r.average base=landuse cover=elevation.dem output=altitud

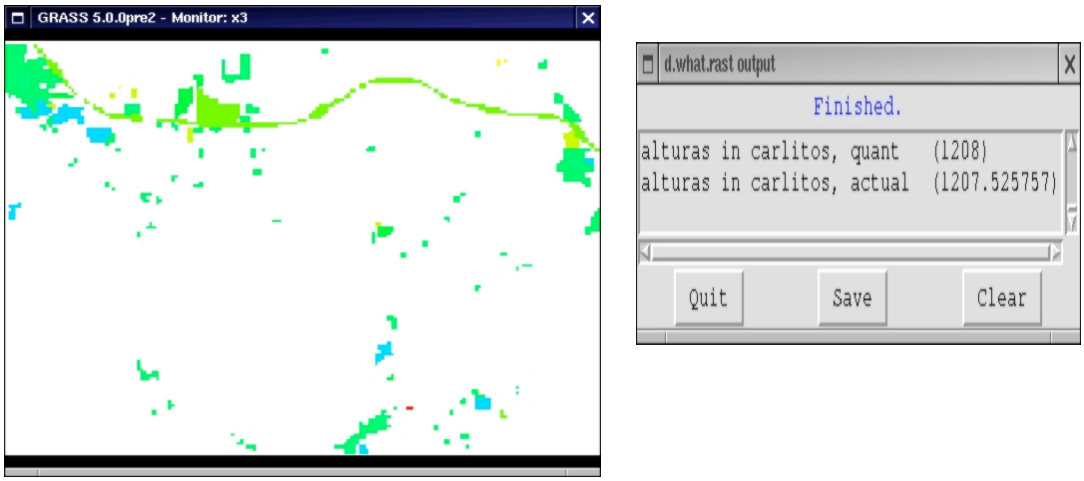


Fig. 5.3. Muestra el mapa resultante del análisis con r.average y los datos de un punto del mapa

Para el tema de análisis estadísticos tenemos un comando más potente que es el r.statistics, pudiendo calcular desviaciones típicas, medias, modas,... Tiene el mismo fundamento que el comando anterior.

Por ultimo veremos el comando más completo que tiene GRASS que es el comando para realizar álgebras de mapas (sumas restas, divisiones,etc), este comando se llama r.mapcalc y tiene muchas posibilidades con respecto a sus posibilidades en este curso veremos su sintaxis y unas pequeñas aplicaciones con este comando.

Para lanzar el programa de calculo de mapas debemos teclear dentro de GRASS el comando r.mapcalc y nos colocara directamente en un prompt propio de este programa mapcalc>, caqui podremos hacer todo tipo de operaciones con mapas, para ello lo mejor es un ejemplo; Vamos a simular una inundación del territorio. Para esto utilizaremos un modelo de simulación que consta de una cota máxima de inundación y un mapa de elevaciones, esto provocara que todas las celdas por debajo de esta cota queden inundados,.

Para este modelo utilizaremos como valor de la cota de inundación 1200 metros y el mapa de elevaciones sobre el que vamos ha hacer la simulación es el de elevation.dem.

Utilizaremos el comando r.mapcalc

mapcalc > inundación=if(elevation.dem<=1200)

elevacion1200=if(elevation.dem<=1200,1200,elevation.dem)

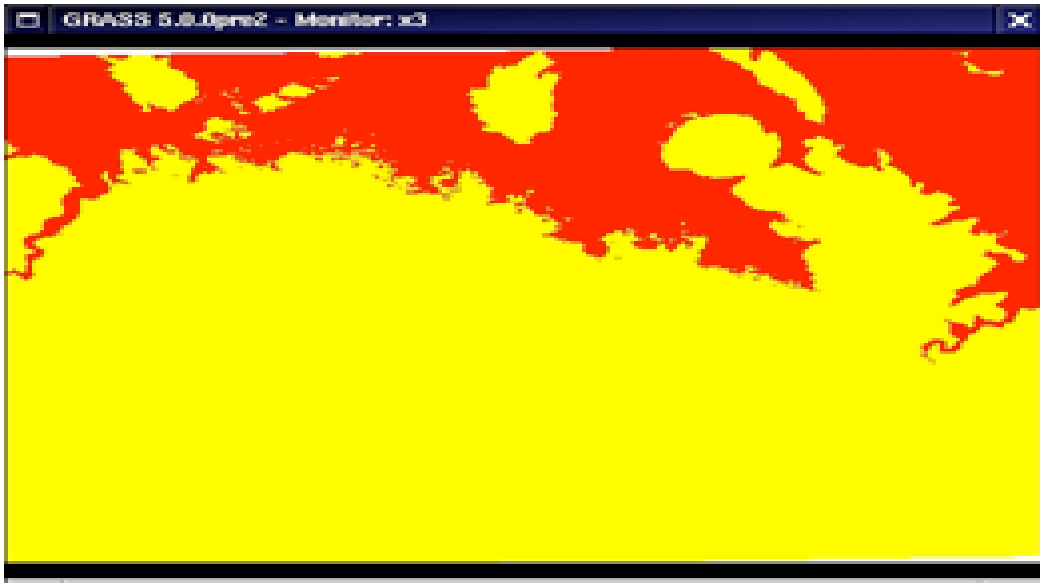


fig 5.4. Calculo de la zona de inundación en mapa elevation.dem

podemos ver las zonas que han sido inundadas en color rojo y en amarillo las que no

ahora podríamos aprovechar que conocemos otros comandos y conocer el área de la zona que ha sido inundada, para ello utilizamos el comando r.report;

r.report inundación units=cel,h

vemos que la zona inundada ocupa un total de 6.630 hectáreas y 73669 pixels

Si nos apetece ya rizar el rizo podemos hacer una representación tridimensional de la inundación.

Ahora nos toca comprobar lo que hemos aprendido en este capitulo

Ejercicios:

1. Vamos a hacer ya un ejercicio para un fin concreto, para ello debemos reiniciar GRASS y seleccionar como LOCATION leics.
2. Calcular el tramo del río que fluye a través del área cultivable e industrial.
3. Calcular el riesgo de una contaminación industrial y seleccionar la población que debido a su situación es más probable que no puedan escapar de esta contaminación.
4. Seleccionar el lugar idóneo donde colocar una planta de aguas residuales en la zona de estudio

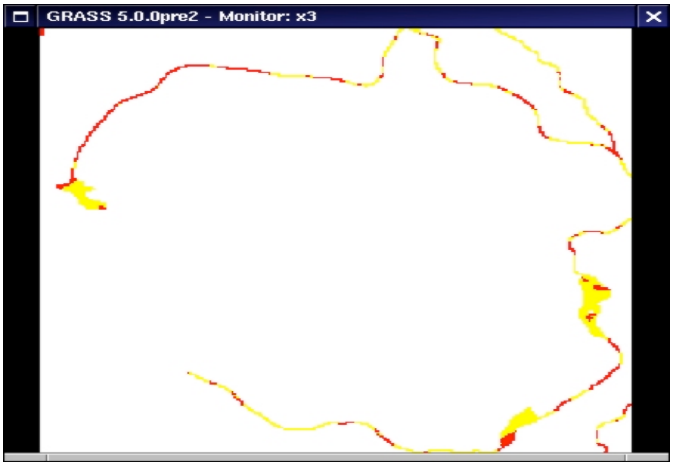
Solución:

1. Primero decir que la zona de estudio leics, que es una localización inglesa en el noroeste de Leicestershire, esta zona tiene una superficie de 12 km2, con diferentes mapas tanto raster, como vectoriales.
2. Para ello lo primero que tenemos que hacer es hacer una reclasificación, reclasificaremos el mapa landcov, seleccionando las capa arable e industrial con valor 1 y las demás con valor 0, con esto tendremos ya un mapa con dos colores; posteriormente vamos a cruzar este mapa con el mapa de water(rios), esto lo hacemos con r.mapcalc.

Mapcalc> rio=arable_industrial*water

veremos un mapa por donde circula el río y en colores las capas que pasa, simplemente a multiplicado celdas y las celdas que no contenían río(0) y las que no eran arables e industriales(0), dan un valor nulo(blanco). Podemos ver la estadística de las celdas que recorre el río e incluso medirlo con el ratón y d.measure.

Fig 5.5. Recorrido del río por las zonas de agricultura eindustrial



3. Para este caso vamos a ver cual puede ser la población que se sitúa más lejos de las vías principales de escape; realizar primero un mapa con las áreas de tierra que se encuentran a una distancia de 1000 metros de las vías de tren, utilizaremos para ello el comando **r.buffer**

r.buffer rail distances=1000 output=trenes_riesgo

también debemos hacer lo mismo con las carreteras, para ello lo mejor sería reclasificar el mapa roads y dejar valores las autopistas(1) y las carreteras normales(0), ahora es el momento de calcular las distancias,

r.buffer=roads distances=1000 output=carreteras_riesgo

aquí podemos ver las zonas que no estarían a salvo de la contaminación por carretera ¡¡Ojo vamos a tener un problema ya que las zonas que nos interesan, es decir las que estan lejos de cualquier tipo de carretera quedan con valor nulo y esto va a crearnos un problema a la hora de realizar el ejercicio, de hecho podemos ver los resultados que dan de una forma y otra. Para hacer este cambio vamos a utilizar brevemente un comando que es **r.support**, es interactivo tecleamos **r.support** y confirmamos los que nos de por defecto hasta que podamos cambiar las las categorías (edit category file) elegimos <y> podemos poner en la categoría 0 maximo peligro, luego continuamos con los las demás preguntas siempre con <enter> hasta que llegue (create/reset null file) donde seleccionamos <y> nos cambiara los valores nulos por valores cero y a la siguiente pregunta respondemos <n> y si ahora vemos nuevamente el mapa y le preguntamos con **d.what.rast** veremos que ya no son valores nulos sino valores cero, esto mismo lo debemos hacer con el mapa trenes_riesgo

Ahora ya tenemos dos mapas de riesgo y debemos sumarlos para conseguir conocer la población que no va a estar afectada por situarse cerca de las vías de comunicación

mapcalc> riesgo_total=trenes_riesgo+carreteras_riesgo

nos sale el mapa de ambas posibilidades juntas, zonas riesgo carreteras(azul), zona riesgo de tren(rojo) y zonas muy alejadas(amarillo), para un mejor calculo vamos a reclasificar el nuevo mapa llamándolo riego_mortal, solo manteniendo dos valores en valor Null para todos los valores ecepto para el valor 0 que tomara valor 1, ahora es el momento de calcular la población contenida en estas zonas alejadas, utilizando los mapas riesgo_mortal y

popln(población),

mapcalc>riesgo_definitivo=riesgo_mortal*popln

con esto ya tenemos en color rojo las poblaciones que tienen posibilidad total de no poder abandonar en una posible contaminación química que viéndolo con **r.report** nos dan 55 celdas tienen maximo peligro en el mapa.

Ahora vamos con la contaminación, desafortunadamente un camión que cruzaba por la zona de estudio tiene un accidente y lleva en su mercancía productos químicos muy contaminantes que pueden causar daños, a la poblaciones que se encuesten a 2000 metros de distancia de la zona del accidente, con lo que las autoridades necesitan saber cuales son las vías de escape(tren o autopistas) que pueden utilizar la población para la evacuación

podemos ver el lugar del accidente en el mapa(crash), hacemos un **r.buffer** a una distancia de 2000metros y luego calcular igualmente que lo hicimos antes la población afectada y sus medios de escape.

4. Por ultimo vamos a ver cual es el mejor sitio para colocar una planta de aguas residuales en nuestra zona.

Lo primero que vamos a hacer es cambiar la región de trabajo, modificándola a

N 316000

S 310000

W 444000

E 450000

resolución 50

una vez que hemos seleccionado una subregión debemos borrar el monitor para que se adapte a esta nueva región con **d.frame -e**, luego veremos los mapas image y con la opción de superponer mapas los mapas source y plant, con esta visualización veremos una fotografía aérea con dos puntos rosados que nos muestran los lugares de la planta residual y de la fuente donde se producen las aguas residuales(source) y donde se sitúa la planta de tratamiento(plant), podemos con las ordenes que ya conocemos saber las coordenadas que tienen esos puntos en el mapa referenciado. El siguiente paso para el análisis es calcular el

mapa de pendientes de la zona, para poder ver los mejores lugares para colocar las tuberías y la colocación de un posible equipo de bombeo, para calcular el mapa de pendientes utilizamos el comando r.slope, con la siguiente sintaxis;

r.slope.aspect elevation=topo aspect=pendientes1

con ello creamos el mapa de pendientes llamado pendientes1 utilizando como mapa base el de topo, ahora es el momento de ver el resultado con d.rast pendientes1, incluso le podemos cambiar los colores para poder ver mejor las pendientes con r.colors, e incluso para rematar podemos cambiar los colores y las intensidades con otro nuevo comando que es d.his(combina el color de un mapa y el sombreado de un mapa aspect(pendientes), como ejemplo podemos hacer;

d.his h_map=landcov i_map=pendientes1

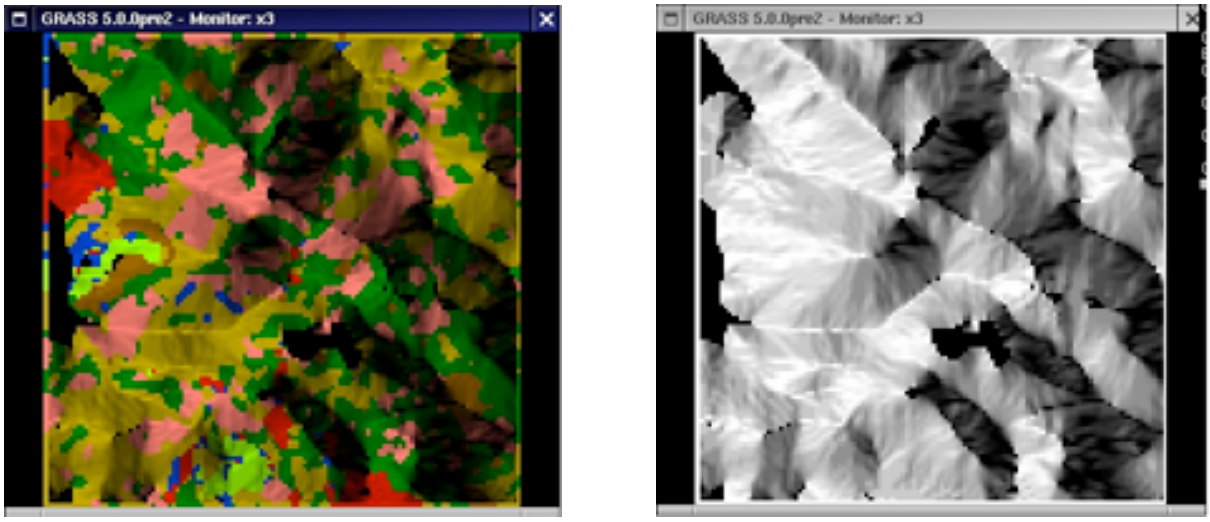


fig.5.6 diferencia entre el coloreado de un mapa de pendientes y el original en escala de grises

Para continuar con el ejercicio vamos a tratar de calcular el coste debido a la dirección de la pendiente a la hora de calcular por donde ira la tubería desde el punto de salida de las aguas hasta la planta de tratamiento, para ello combinamos los comando r.rescale y r.mapcalc, primero reclasificaremos el mapa de pendientes1

r.rescale input=pendientes1 from= 0,180 output=pend_norte to=10,10

r.rescale input=pendientes1 from= 181,360 output=pend_sur to=1,1

no sale nada y esto es debido a que los datos que no contamos en cada cara los pone nulos y

sale un mapa blanco en las zonas de pend_sur y viceversa, podemos probarlo al sumar con r.mapcalc;

mapcalc>pend_coste=pend_norte+pend_sur

con esto lo que pretenderíamos es tener un mapa con dos colores, uno que fuera valor 10 donde estuvieran las pendientes cara norte las que mas nos costaría lanzar la tubería y valor 1 las de la cara sur que menos costaría llevar la tubería, con respecto principalmente a la presión, el problema es que si lo hacemos de esta forma nos sale un mapa nulo, pero existe una solución

r.rescale input=pendientes1 from=0,360 output=pend_coste to=0,1

con esto obtenemos un mapa de pendientes pero lo mejor es reclasificarlo a nuestro gusto para poder poner a la parte de pendiente el valor que queramos, si utilizamos directamente los valores que queremos o sea en to=1,10, vamos a tener un problema ya que lo que vamos a obtener es una gama de colores desde 0 a 360 grados, primero vamos a reclasificar el mapa pend_coste, pero seguramente no lo va a permitir ya que es una reclasificación de otra reclasificación, para ello lo mejor es crear otro mapa igual pero sin las uniones, es decir con mapcalc indicamos pend_coste=pend_coste2, y luego **r.reclass**, ahí es donde podemos modificar las cuadriculas a nuestro antojo lo llamaremos pendiente_coste, ya podemos poner el valor 10 para las pendientes 0 a 180 y 1 para las de 180 a 360, este clasificación va en función desde donde vayamos a considerar el inicio de la tubería ya que si calculamos el camino desde la planta hasta la salida de aguas el planteamiento de coste sería el contrario.

El siguiente paso es calcular el coste debido a los usos del suelo de la zona, para ello utilizamos **r.reclass** con el mapa landcov y lo llamamos lan_cost.

Principalmente lo que vamos a hacer es reclasificar las capas que tenemos en el mapa para poner los valores máximos a las capas por las que tiene mayor coste el transporte de la tubería o es más peligroso, así podemos ver en la figura siguiente como los valores máximos los tiene n la capa agua y la capa cantera y así con las demás capas hasta dar valor 1 a la capa de pastizal. Podemos cambiar el color de las capas a nuestro gusto lo mejor es con el comando **d.colors**, este tiene un menú interactivo donde podemos ir eligiendo las intensidades y mezclas de color de cada capa.

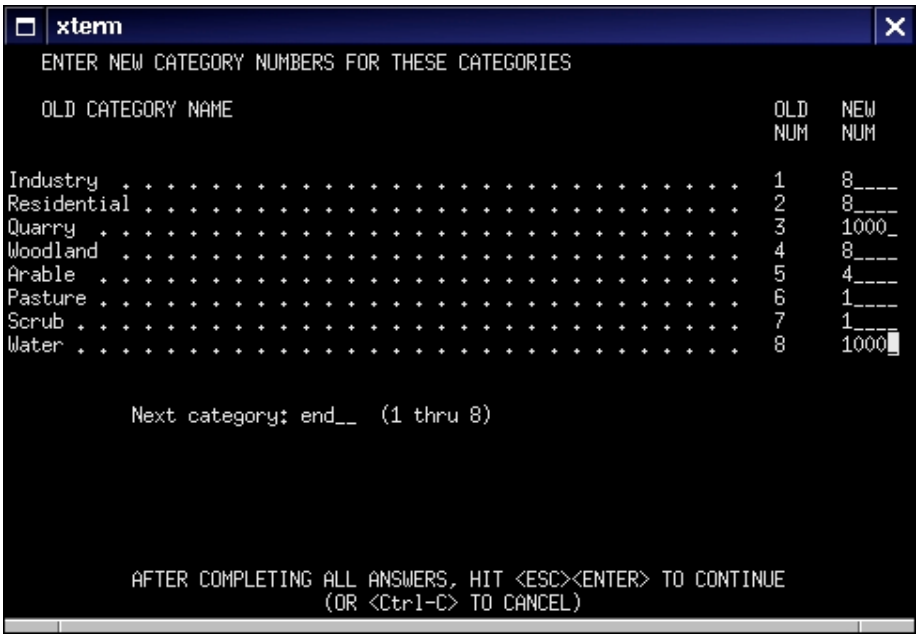


Fig. 5.7 Nueva reclasificación del mapa de cobertura (landcov)

Es el mejor momento para calcular el coste final asociado con cada celda combinando los mapas de coste de pendiente (pendiente_coste) y el mapa del coste de cobertura lan_cost, utilizamos el álgebra de mapas **r.mapcalc**.

Mapcalc> coste=pendiente_coste+lan_cost

al ver el mapa resultante nos ocurre lo mismo que antes parece que sólo hay dos categorías de colores pero existen más, ahora lo vamos a solucionar de otra manera que es convirtiendo esa escala de colores a escala de grises, utilizamos el comando **i.grey.scale** e indicamos nuestro mapa coste, y ya podemos ver el mapa con todas las categorías de colores que indica el coste de llevar la tubería por cada cuadricula del mapa de la zona de estudio.

Ahora vamos a calcular la distancia y su coste por la superficie para la tubería, vamos a calcular lo que costaría meter la tubería desde el punto de salida de las aguas residuales hasta la planta, calculado a través del paso por cada celda, usaremos para este calculo un nuevo comando que es el comando **r.cost** este comando permite calcular el coste acumulativo que se produce al moverse entre diferentes localizaciones geográficas, para usar este comando debemos dar las coordenadas geográficas del punto desde donde queremos comenzar, vamos a ver para nuestro ejercicio, utilizamos como punto de inicio el de la salida de aguas residuales, como hemos visto antes si consideramos como punto de partida la planta debemos introducir otras coordenadas.

r.cost input=coste coord=444525,313875 output=coste_distancia

hemos seleccionado la opción coordenada para esta orden indicando la posición del Este,Norte del punto de inicio

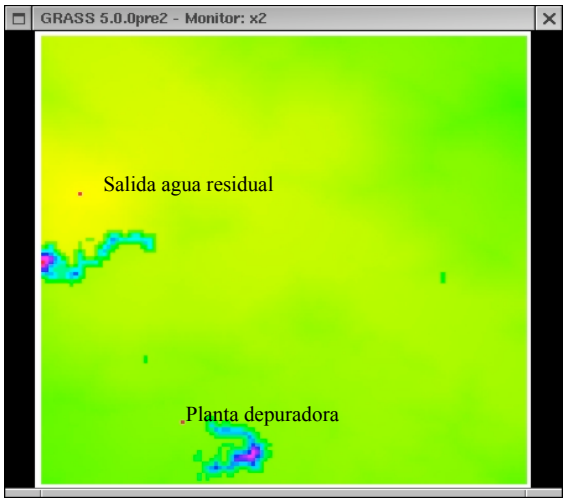
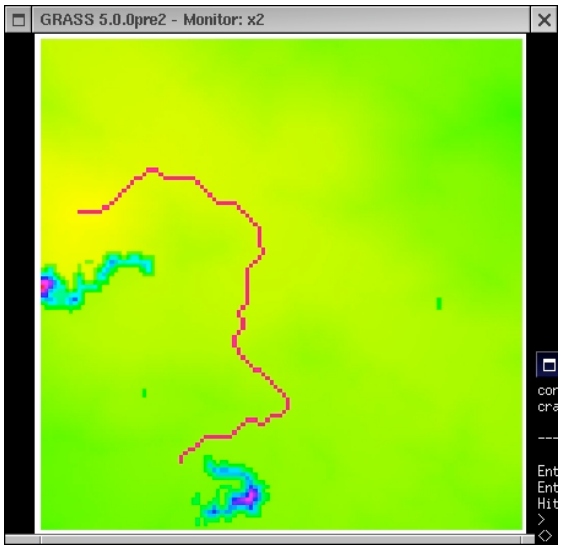


fig. 5.8 Calculo del coste desde las salidas del agua residual.

Calcularemos la ruta de menor coste para meter la **tuberia**, esto lo podemos hacer con el comando **r.drain**(sumidero), sólo necesitamos poner el mapa de coste calculado anteriormente e indicar las coordenadas del punto destino que en nuestro caso es la planta depuradora.

r.drain input=coste_distancia coord=445775,310875 output=tubería



Podemos observar como el programa calcula el mejor camino y de menor coste para llevar la tubería desde la salida de las aguas residuales hasta la entrada de las aguas en la planta depuradora.

Fig. 5.9. Ruta de colocación de la tuberia

Ya hemos resuelto el ejercicio pero ahora vamos a realizar algunas modificaciones al mapa para que tenga un mejor presentación, lo primero que vamos a hacer es modificar los colores del mapa para que se parezcan más a las categorías de terreno que tiene la zona de estudio, reclasificamos los valores con **r.reclass**, la reclasificamos de su valor inicial que era 1 y ahora le ponemos el valor 9. Vamos a utilizar otro comando que es **r.patch**(comando que crea una composición de mapas raster usando categorías conocidas), vamos a comprobarlo

r.patch tubería2,landcov output=tubería_final

d.his h_map=tuberia_final i_map=pendientes

obtenemos un mapa muy atractivo, e incluso podemos verlo entres dimensiones utilizando como mapa de elevaciones el mapa topo.

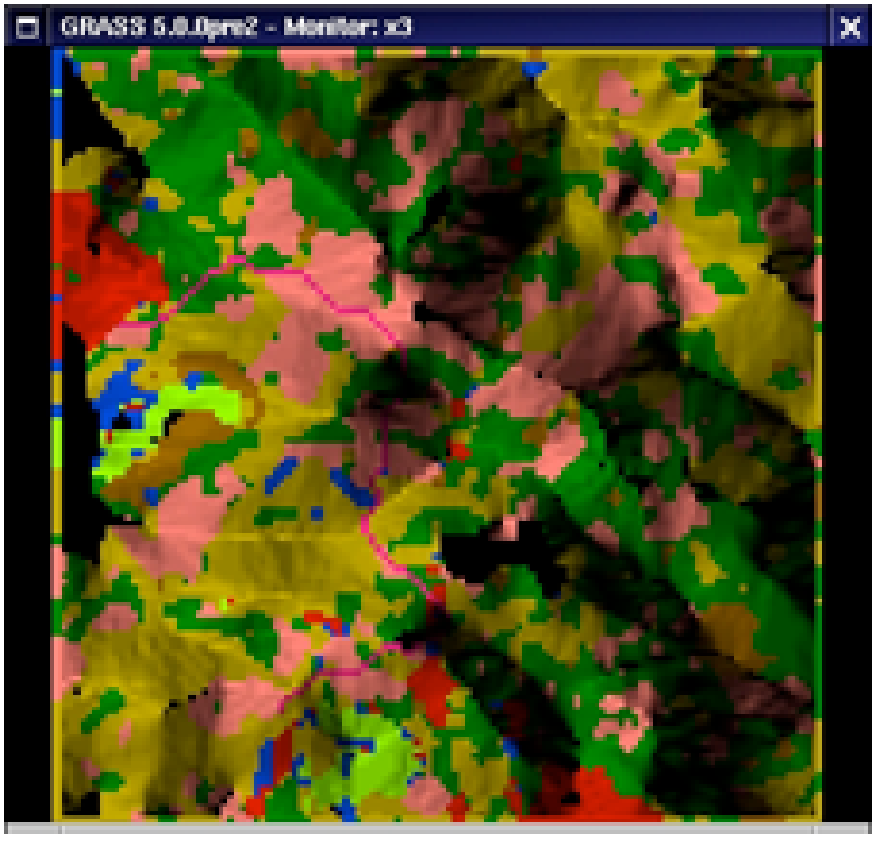


Fig. 5.10 Visión del mapa resultado del ejercicio

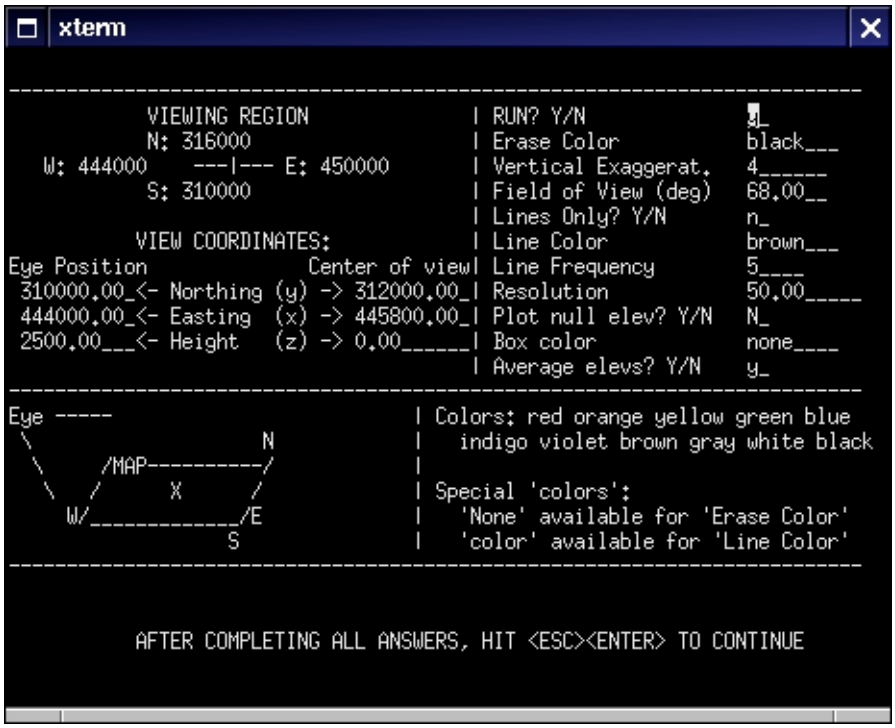


fig. 5.11. Datos para una correcta visualización del mapa en 3D

Con los datos apropiados para ver mejor la vista en tres dimensiones, y la colocación del punto de vista tenemos la siguiente visión 3d del ejercicio

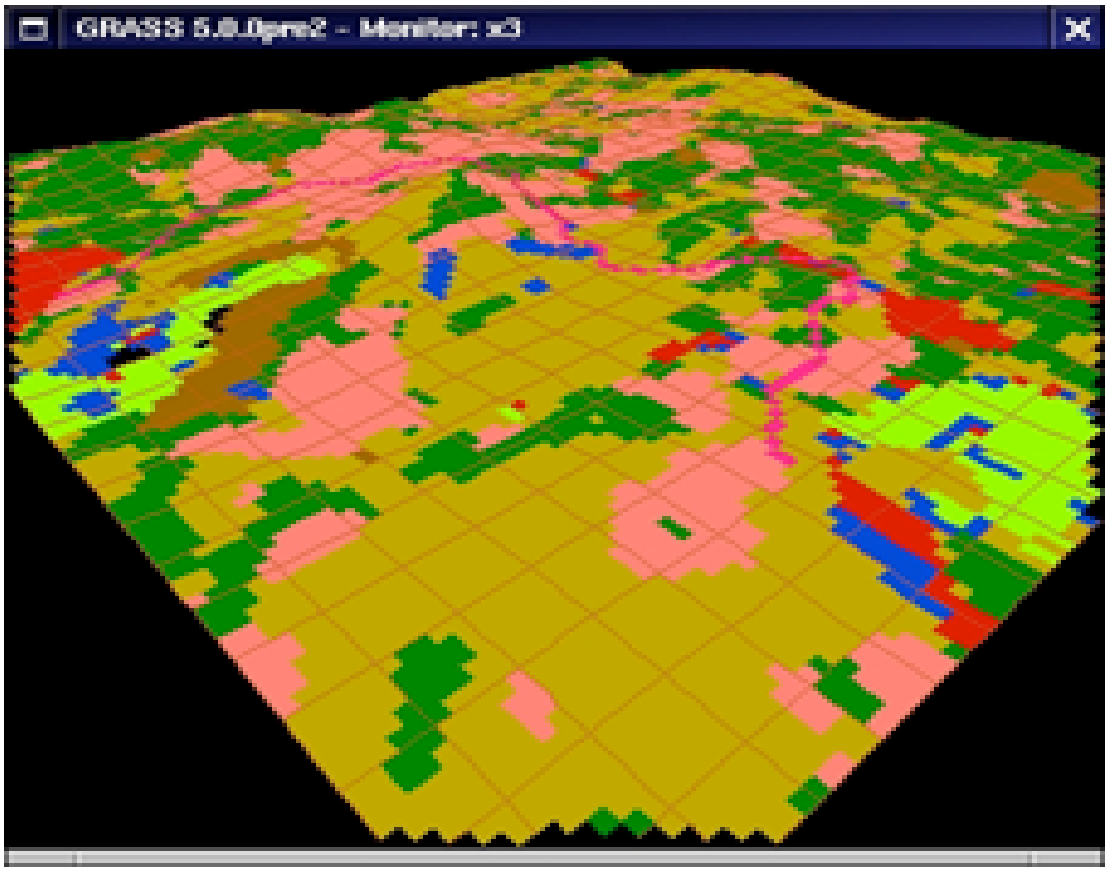


Fig. 5.12 Visualización del resultado del ejercicio en modelo 3D

Con esto finalizamos este capitulo muy completito.

Capitulo VI: Importando mapas

Esta capitulo principalmente tratara de mostrarnos las herramientas de importación de mapas que posee GRASS para poder ver y manipular mapas de otros sistemas externos

1. Conocimiento de los principales comandos que posee GRASS para importar datos, mapas y fotos

Manejaremos diferentes comandos de GRASS como son:

- **r.in.ascii**: Convierte un mapa binario en formato raster
- **r.in.erdas:** Crea un mapa raster a partir de un mapa en formato de ERDAS
- **r.in.gif**: Convierte un fichero en formato GIF(de 8bit) en formato raster
- **r.in.ll**: Convierte un fichero raster en un mapa raster referenciado
- **r.in.poly**: Crea un mapa raster a partir de un fichero ASCII que tenga un poligono o una linea
- **r.in.tiff**: Convierte una foto de formato TIFF a un mapa formato raster
- **r.in.arc**: Convierte ficheros generados por arc view y arc info en formato GRASS
- **m.in.e00**: Permite leer y pasar a formato vectorial un mapa elaborado por arc info
- **v.to.rast**: Convierte un mapa vectorial en un mapa raster
- **v.import**: Permite interactivamente hacer importación de mapas vectoriales generados en cualquier plataforma y tipo de archivo.

Empezamos con los distintos comandos de importación:

- **r.in.ascii**: Asociado a este comando tenemos el contrario que es el que nos permite exportar un mapa raster a un fichero binario, es el comando **r.out.ascii**. Con este comando podemos realmente conocer mejor el funcionamiento de los mapas en formato raster, lo que conseguimos con este comando es poder ver en formato texto las características que posee el mapa tanto el valor de las coordenadas del lugar, el numero de filas y columnas e incluso el valor de cada celda, podemos ver un ejemplo de un mapa en cuestión, vamos a ver el contenido del mapa de landcov(que esta dentro de leics).

r.out.ascii input=landcov output=suelo

este fichero creado lo podemos ver con cualquier tipo de editor de texto(vi,emac,joe,pico,...), podemos ver que tiene una estructura de numeros que indica perfectamente los valores de cada celda. Ahora podemos utilizar el comando de importación para convertir este fichero de texto el cual podemos modificar, podemos probar a cambiar una serie de valores de los pixels y ver como cambia el mapa raster al importarlo otra vez a nuestra LOCATION.

-**r.in.erdas**: podemos crear mapas raster a partir de mapas que tengamos en el formato ERDAS, este comando sera útil para las personas que hayan utilizado ERDAS en nuestro caso no vamos a profundizar en este comando.

-**r.in.poly**: Es un programita que crea un mapa raster a partir de un fichero vectorial de lineas o áreas en ASCII. Para probar el funcionamiento de este comando haremos una prueba editando un fichero para crear un cuadrado, para ello lanzamos un editor de texto sencillote (pico,nano,vi,emac,joe,...), en este caso voy a utilizar joe, y edito la siguiente información:

fig 6.1. Creación de un cuadrado a partir de la edición de sus vértices en modo texto

-**r.in.gif**: Convierte formatos GIFF en raster, este tipo de importación la trataremos con imágenes en TIFF

-**r.in.tiff**: Convierte imágenes en formato TIFF, por ejemplo una foto aérea escaneada a formato raster, en este caso debemos convertir una fotografía obtenida desde internet a formato raster, podemos buscar cualquier imagen del satélite spot 5.

-**r.in.arc**: Convierte ficheros generados con arc info en formato GRID a raster

-**m.in.e00**: Podemos convertir formatos de arc info a GRASS, solo necesitamos seguir las indicaciones de este comando interactivo para ver los resultados en un mapa vectorial con el comando d.vect, ya solo nos queda permitir que cada uno pruebe con los distintos comandos de importación para GRASS y con ficheros que posean en otros formatos, este capitulo es a

modo de introducción para que los usuarios que estén pensando en cambiar a GRASS sepan que existen comandos que permiten trabajar con sus mapas generados con otros sistemas de información geográfica comerciales.

-**v.to.rast**: Con este comando podemos rasterizar mapas que estan en formato vectorial, como ejemplo vamos a pasar el mapa vectorial roads(de spearfish) a un mapa raster llamado carreteras

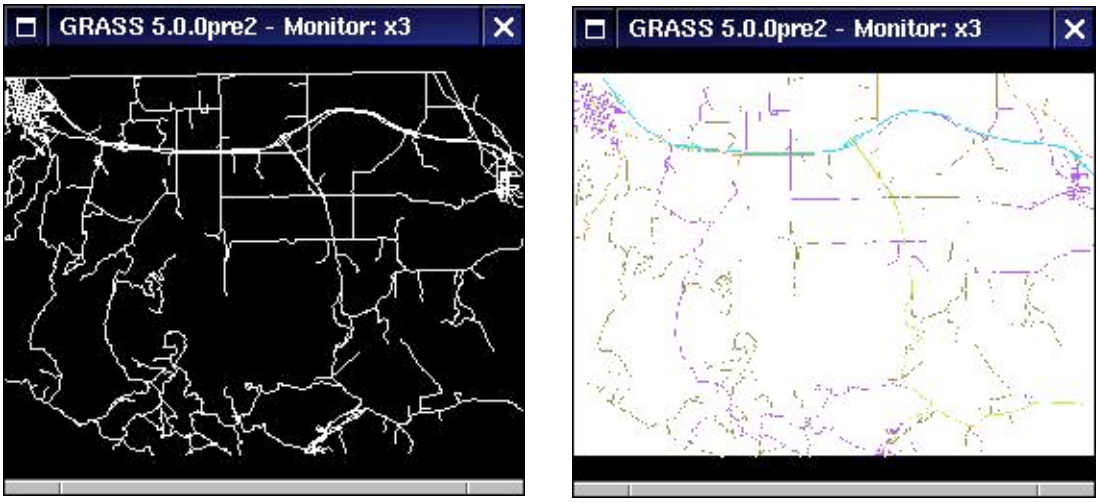


fig. 6.2. Distinto formato mapa en vectorial y transformación de este a raster

-**v.import**: Permite mediante un menú interactivo importar los estilos de ficheros que nosotros necesitemos ver en GRASS

Todos estos comandos los tenemos en el entorno gráfico en el menú principal en import

COMENTARIO FINAL

Aquí finaliza el manual de Introducción a Grass, elaborado por Carlos Campillo Torres (perteneciente al área de Ecología del departamento de Física de la Universidad de Extremadura). Este manual ha tenido la finalidad de introducir al usuario en un sistema de información geográfica de visualización y generación de mapas libre y que su formato corre sobre Linex

Agradecer a todos los que han ayudado a realizar este manual y especialmente a:
Teresa Buyolo, José Cabezas, José Martín, Daniel Paton, Fátima Sanz y Pedro Reyes Testeadores de este manual además de todas las personas que trabajan por que este sistema de información geográfica avanzado, seguro y libre, especialmente al grupo de trabajo de la Universidad de Murcia que lleva varios años ya trabajando con este sistema y que fueron los que me introdujeron en el conocimiento de este programa.

El material de apoyo utilizado para este manual fue:

- GRASS SEEDS: A Beginner tutorial Langford M. and Wood J.
- Apuntes de Practicas de la asignatura (SIG y Teledetección) Universidad de Murcia Francisco Alonso Sarria
- Apuntes Curso GRASS, impartido por José Antonio Palazón
- Manual de GRASS online
- Localizaciones de ejemplo de LEICS y SPEARFISH pagina web http://grass.itc.it/data.html

NOTA: Esta es la versión 1.1 del manual de Introducción a GRASS, para cualquier tipo de sugerencia o corrección podéis poneros en contacto con el autor en el correo ccampy@unex.es