

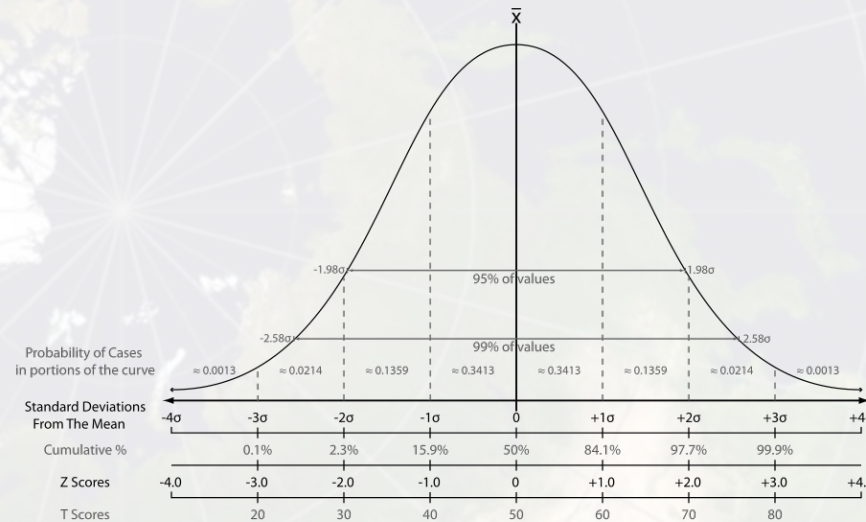
GRASS GIS 7: A theoretical and practical course

Session 1 – GRASS GIS 7 and R-stats intro

Markus Neteler

NINA 2016, Oslo, Norway

mundialis GmbH & Co. KG
<http://www.mundialis.de>





Session Objectives

- GRASS GIS 7 and R-stats intro
- Examples:
 - Using R within a GRASS GIS session (incl RStudio)
 - Relationship elevation, latitude, distance from sea and temperature
(this is a “circular” analysis, trying to reconstruct the input data from random sampling)



Exercise: GRASS GIS 7 and R statistics

GRASS GIS 7.0.2svn startup (r65855)

 **grass gis**
Bringing advanced geospatial technologies to the world

1. Select GRASS GIS database directory

GRASS GIS database directory contains Locations.

2. Select GRASS Location

demolocation
ecad5_grassdata_ll
nc_spm_08_grass7
nc_spm_08_reduced
spearfish70

All data in one Location is in the same coordinate reference system (projection). One Location can be one project. Location contains Mapsets.

3. Select GRASS Mapset

PERMANENT
user1

Mapset contains GIS data related to one project, task within one project, subregion or user.



Exercise: GRASS GIS 7 and R statistics

Using R within a GRASS GIS session

```
grass72
```

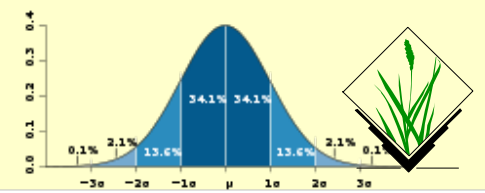
Set computational region to the map:

```
g.region raster=elev_state_500m -p
```

(Start of R – see next slide)

Wiki tutorial:

http://grass.osgeo.org/wiki/R_statistics → rgrass7 subpage



GRASS-R interface

Using R in a GRASS GIS session (North Carolina location)

R on Linux:

Start R from GRASS session:

```
GRASS 7 > R
```

... or: start rstudio from GRASS session:

```
GRASS 7 > rstudio
```

R on Mac OSX:

```
GRASS 7 > /Applications/RStudio.app/Contents/MacOS/RStudio
```

R in MS-Windows:

- use the “Explorer” to find the “RStudio.exe” in the R binaries folder
- **swipe** its **file icon** into the terminal of the running GRASS session, eg:

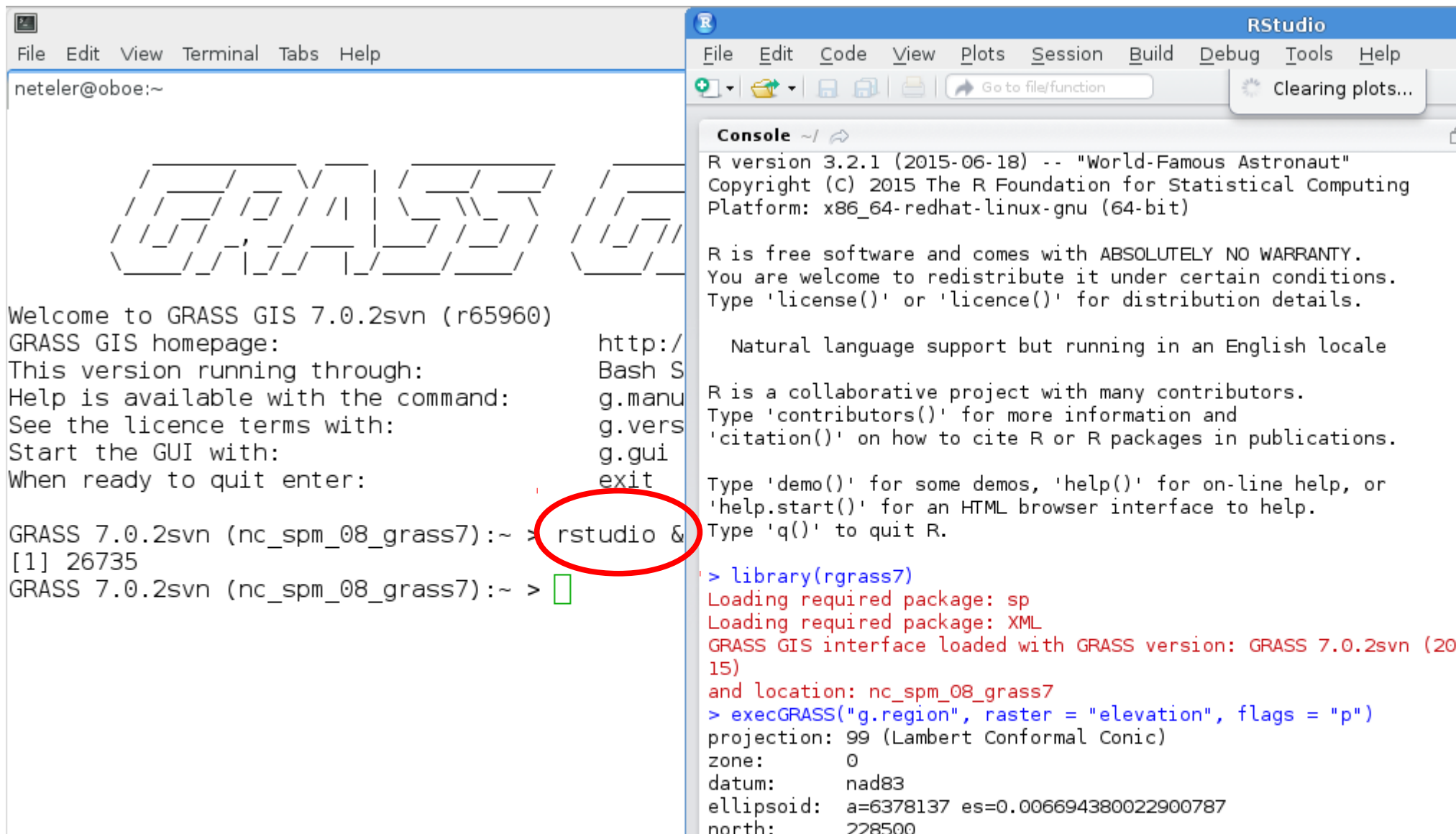
```
C:\Program Files\RStudio\bin\rstudio.exe
```



Exercise: GRASS GIS 7 and R statistics

Using R within a GRASS GIS session

https://grasswiki.osgeo.org/wiki/R_statistics/rgrass7



```
neteler@ofoe:~  
  
Welcome to GRASS GIS 7.0.2svn (r65960)  
GRASS GIS homepage: http://grass.osgeo.org/  
This version running through: Bash Shell  
Help is available with the command: g.manual  
See the licence terms with: g.version  
Start the GUI with: g.gui  
When ready to quit enter: exit  
  
GRASS 7.0.2svn (nc_spm_08_grass7):~ > rstudio &  
[1] 26735  
GRASS 7.0.2svn (nc_spm_08_grass7):~ >   

```

```
RStudio  
File Edit Code View Plots Session Build Debug Tools Help  
Go to file/function  
Clearing plots...  
  
Console ~/  
R version 3.2.1 (2015-06-18) -- "World-Famous Astronaut"  
Copyright (C) 2015 The R Foundation for Statistical Computing  
Platform: x86_64-redhat-linux-gnu (64-bit)  
  
R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.  
  
Natural language support but running in an English locale  
  
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.  
  
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.  
  
> library(rgrass7)  
Loading required package: sp  
Loading required package: XML  
GRASS GIS interface loaded with GRASS version: GRASS 7.0.2svn (2015)  
and location: nc_spm_08_grass7  
> execGRASS("g.region", raster = "elevation", flags = "p")  
projection: 99 (Lambert Conformal Conic)  
zone: 0  
datum: nad83  
ellipsoid: a=6378137 es=0.006694380022900787  
north: 228500
```

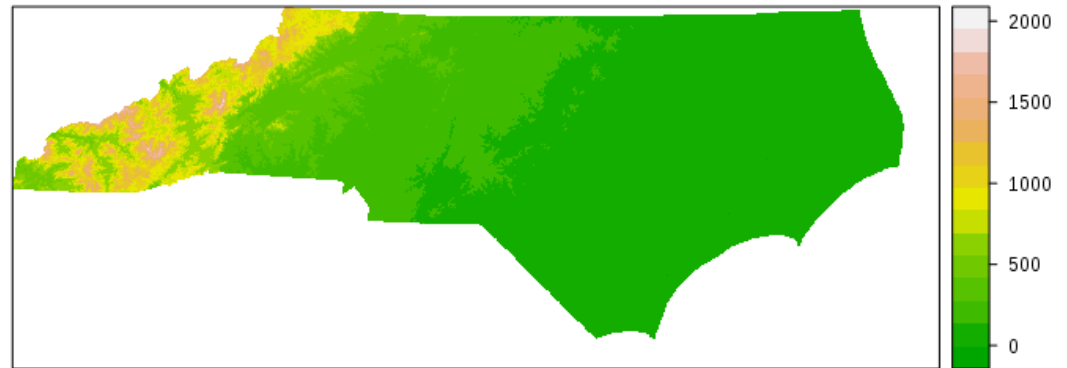


Exercise: GRASS GIS 7 and R statistics

GRASS GIS and R statistical language (cont'd)

```
# get online help:  
?readRAST
```

```
# get full help:  
help.start()
```



```
# set computational region  
execGRASS("g.region", parameters =  
  list(raster = "elev_state_500m"), flags = "p")
```

```
# load GRASS DEM into R:  
elev <- readRAST("elev_state_500m")
```

```
# show map metadata:  
summary(elev)
```

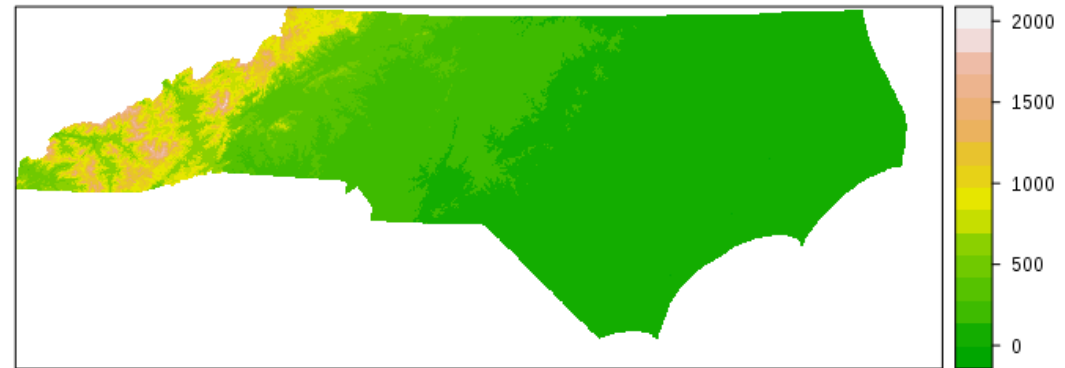
```
# show map:  
spplot(elev, col.regions=terrain.colors(20))
```



Exercise: GRASS GIS 7 and R statistics

GRASS GIS and R statistical language (cont'd)

```
# metadata
gmeta()
gisdbase      /home/neteler/grassdata
location      nc_spm_08_grass7
mapset        user1
rows          669
columns       1678
north         318500
south         -16000
west          124000
east          963000
nsres         500
ewres         500
projection     +proj=lcc +lat_1=36.166666666666666
               +lat_2=34.333333333333334 +lat_0=33.75 +lon_0=-79
               +x_0=609601.22 +y_0=0 +no_defs +a=6378137
               +rf=298.257222101 +towgs84=0.000,0.000,0.000 +to_meter=1
```





Exercise: GRASS GIS 7 and R statistics

Using R within a GRASS GIS session (cont'ed)

```
# list available vector maps:
> execGRASS("g.list", parameters = list(type = "vector"))
# loading a vector map into R space:
> myroads <- readVECT("roadsmajor")
Exporting 355 features [...]
It has 7 fields [...]

> str(myroads)
Formal class 'SpatialLinesDataFrame' [package "sp"] with 4 slots
 ..@ data      : 'data.frame':  355 obs. of  7 variables:
 .. ..$ cat      : int [1:355] 1 2 3 4 5 6 7 8 9 10 ...
 .. ..$ MAJORRDS_ : num [1:355] 1 2 3 4 5 6 7 8 9 10 ...
 .. ..$ ROAD_NAME: Factor w/ 16 levels "I-40","I-440",...: 7
 .. ..$ MULTILANE: Factor w/ 2 levels "no","yes": 1 1 1 1 1
 .. ..$ PROPYEAR  : int [1:355] 0 0 0 0 0 0 0 0 0 0 ...
 .. ..$ OBJECTID  : int [1:355] 1 2 3 4 5 6 7 8 9 10 ...
 .. ..$ SHAPE_LEN: num [1:355] 4825 14393 3213 13392 7196 .
 ..@ lines      :List of 355
 .. ..$ :Formal class 'Lines' [package "sp"] with 2 slots
 .. .. ..@ Lines:List of 1
```



Exercise: GRASS GIS 7 and R statistics

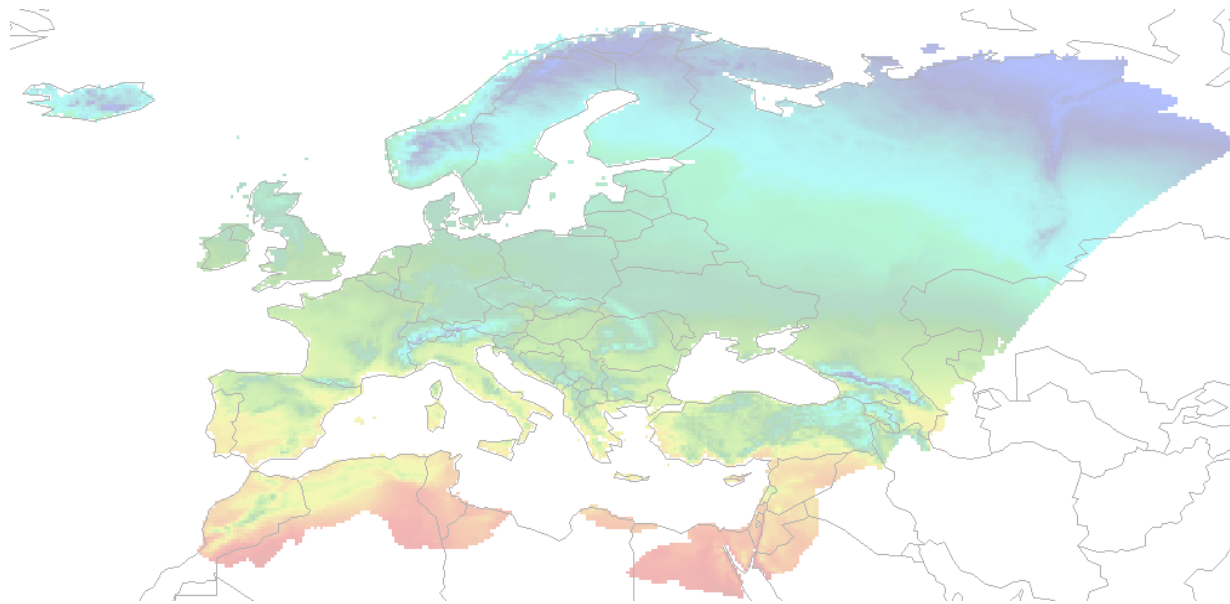
Using R within a GRASS GIS session (cont'ed)

```
> summary(myroads)
Object of class SpatialLinesDataFrame
Coordinates:
      min      max
x 611136.5 676800.5
y 197465.3 257970.1
Is projected: TRUE
proj4string :
[+proj=lcc +lat_1=36.166666666666666 +lat_2=34.333333333333334
+lat_0=33.75 +lon_0=-79 +x_0=609601.22 +y_0=0 +datum=NAD83 +
+no_defs +ellps=GRS80 +towgs84=0,0,0]
Data attributes:
      cat      MAJORRDS_      ROAD_NAME      MULTILANE
Min.    : 1.0    Min.    : 1.0    US-64    : 14    no    :115    Mi
1st Qu.: 89.5    1st Qu.: 91.5    US-401   : 13    yes   :154    1s
...

# end of session
> q()
```



Statistics with GRASS GIS and R: Analysing ECA&D climatic data





Overview

Course data download address:

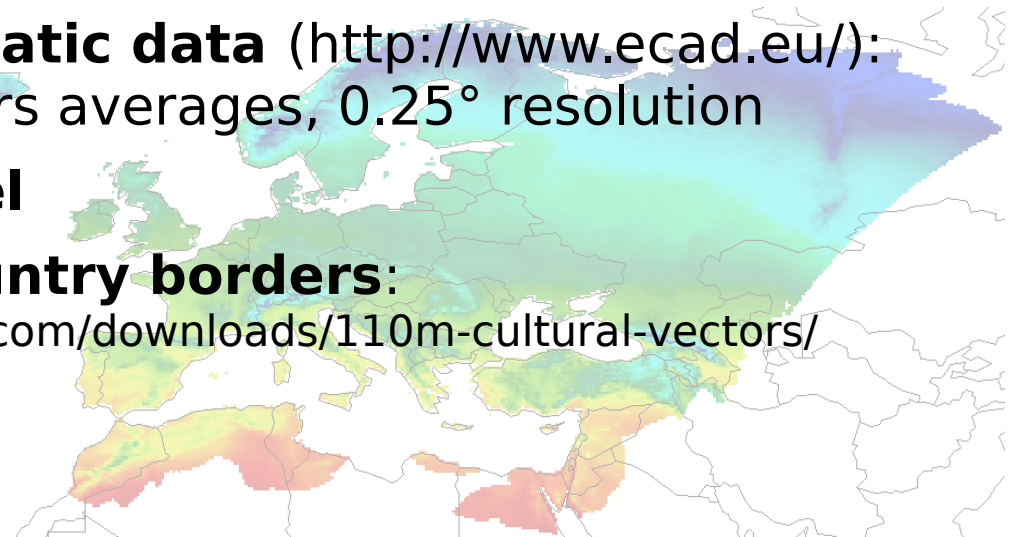
http://www.mundialis.de/workshops/nina2016/ecad5_geotiffs/

- `ecad5_geotiffs_LL.zip|tar.gz` 5.1MB

*Get the package and save it on your computer.
Unpack the GeoTIFF files into your HOME/gis_data/*

Content for exercises:

- Aggregated **ECA&D climatic data** (<http://www.ecad.eu/>):
monthly data of 30 years averages, 0.25° resolution
- Related **elevation model**
- NaturalEarth Admin0 **country borders**:
<http://www.naturalearthdata.com/downloads/110m-cultural-vectors/>





Overview

Aggregated daily ECA&D data in detail:

elevation model (DEM):

ecad5_geotiffs_LL/elev_0.25deg_reg_v6.0.tif

12 maps of monthly sums over 30 years

ecad5_geotiffs_LL/precip_1951_1980_monthly_sums.tif

ecad5_geotiffs_LL/precip_1981_2010_monthly_sums.tif

average of annual sums over 30 years

ecad5_geotiffs_LL/precip_1981_2010_avg.tif

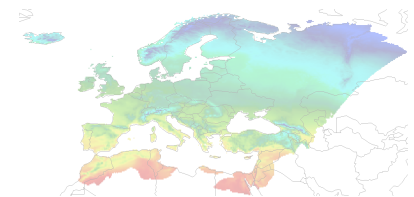
12 maps of monthly tmean averaged over 30 years

ecad5_geotiffs_LL/tmean_1951_1980_monthly_avgs.tif

ecad5_geotiffs_LL/tmean_1981_2010_monthly_avgs.tif

Solar angles calculated from elev_0.25deg_reg_v6.0.tif in GRASS GIS:

ecad5_geotiffs_LL/ecad_solarangle_elev.tif



Exercise – GRASS GIS 7 startup with ECA&D data



GRASS GIS 7.0.2svn startup (r65960)

 **grass gis**
Bringing advanced geospatial technologies to the world

1. Select GRASS GIS database directory

GRASS GIS database directory contains Locations.

2. Select GRASS Location

All data in one Location is in the same coordinate reference system (projection). One Location can be one project. Location contains Mapsets.

3. Select GRASS Mapset

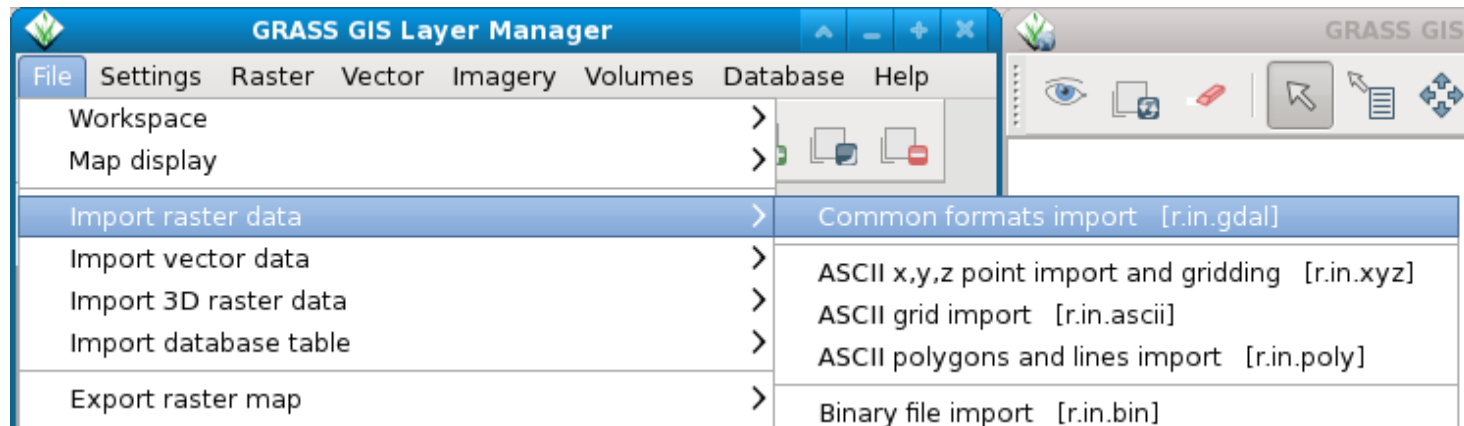
Mapset contains GIS data related to one project, task within one project, subregion or user.



Exercise – Importing data

Import the elevation map in GeoTIFF format

The ECA&D elevation map is the GeoTIFF file
“elev_0.25deg_reg_v6.0.tif”



Exercise – Importing data



Import
elevation
map dialog

We rename to
“elev_ecad”

Import raster data

Settings
Load settings: [dropdown] [Save] [Remove]

Source type
☒ File ☐ Directory ☐ Database ☐ Protocol

Source settings
Format: [GeoTIFF] [dropdown]
File: [/home/neteler/geostat2012_muenster/data/ecad5_geotiffs_LL/el] [Browse]

List of GDAL layers

Layer id	Layer name	Name for GRASS map (editable)
☒ 1	elev_0.25deg_reg_v6.0.tif	elev_ecad

(double click to change name)

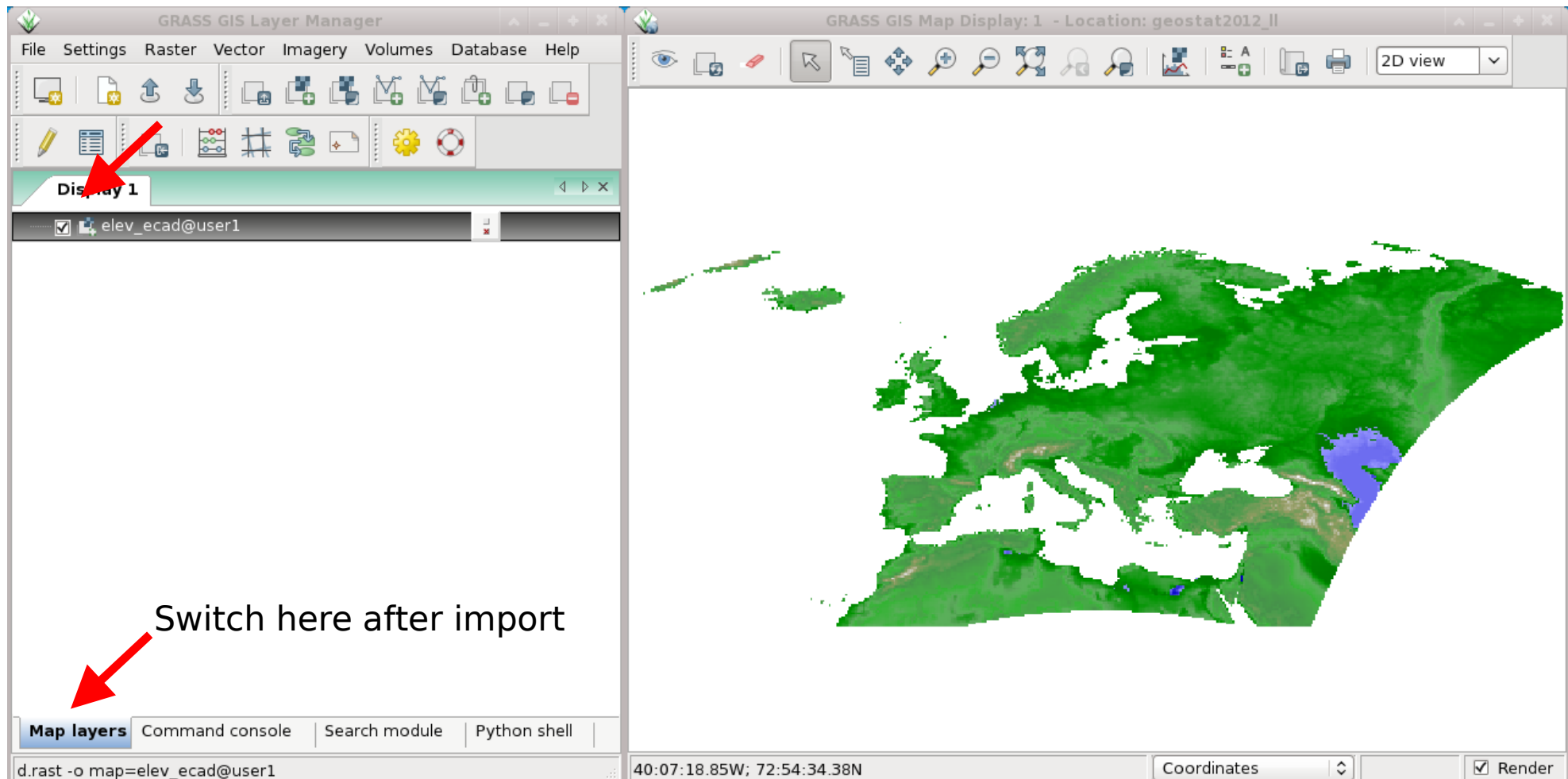
Options
☐ Keep band numbers instead of using band color names
☐ Extend region extents based on new dataset
☐ Force Lat/Lon maps to fit into geographic coordinates (90N,S; 180E,W)
☐ Override projection (use location's projection)
☐ Allow output files to overwrite existing files
☒ Add imported layers into layer tree
☐ Close dialog on finish

[Command dialog] [Close] [Import]

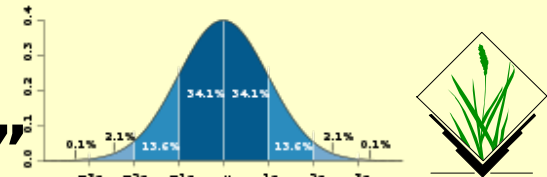


Exercise – GRASS startup and first steps

Click on the checkbox in layer manager to show the imported map.
After import, apply the “terrain”, “elevation” or “srtm” color table.

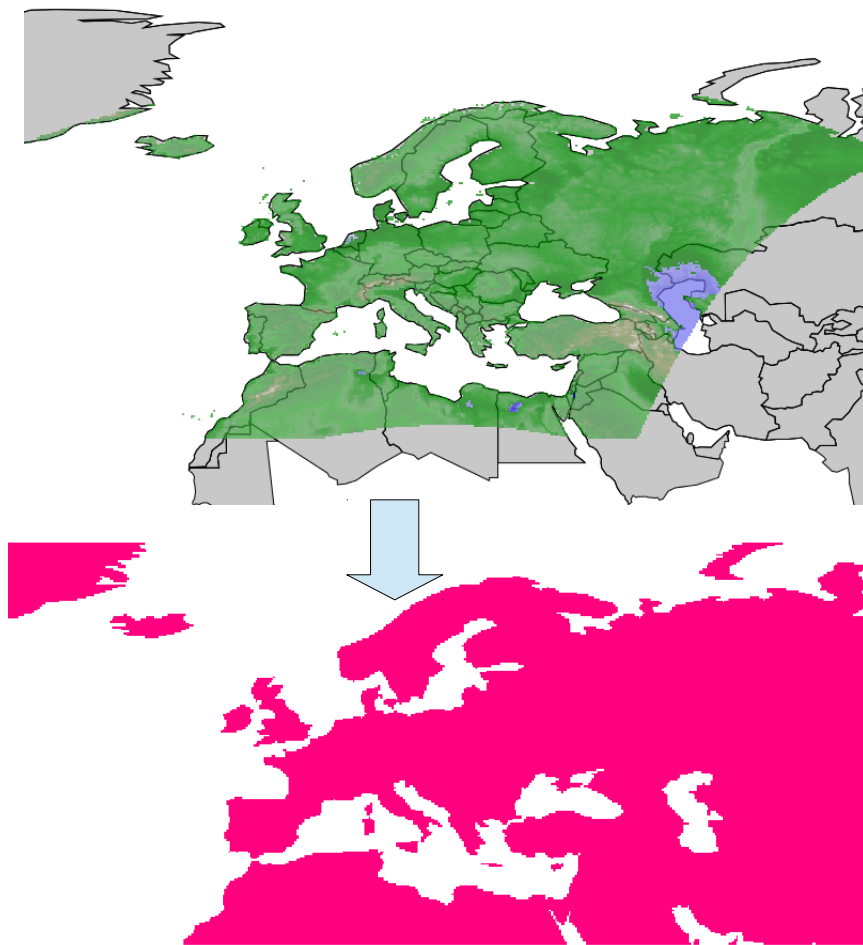


Analysing the relationship between “distance from sea” and “temperature”



```
# As usual, let's be picky about the computational region:  
g.region raster=elev_ecad -p
```

```
# generate a “sea” map first (you may use the country borders):  
v.to.rast in=country_boundaries output=country_boundaries \  
use=val value=1
```



v.to.rast [vector, conversion, raster, rasterization]

Converts (rasterize) a vector map into a raster map.

Required Selection Attributes Optional Command output Manual

Name of input vector map: (input=name)
country_boundaries@PERMANENT

Name for output raster map: (output=name)
country_boundaries

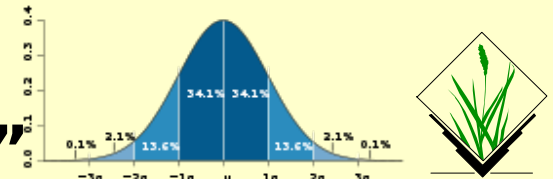
Source of raster values: (use=string)
val

Close Run Copy Help

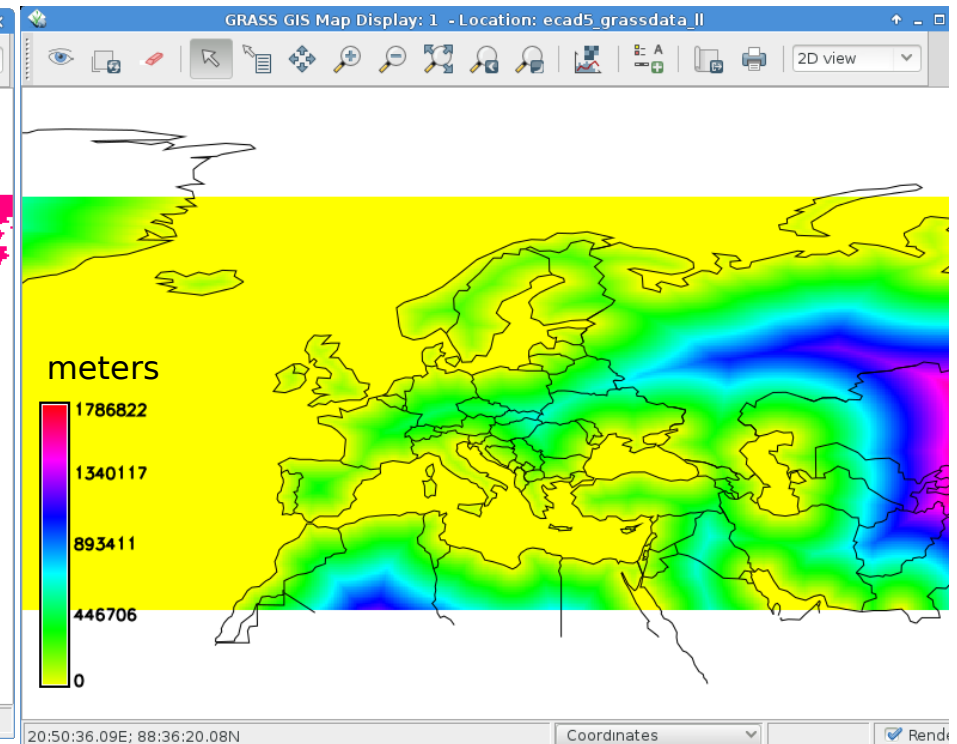
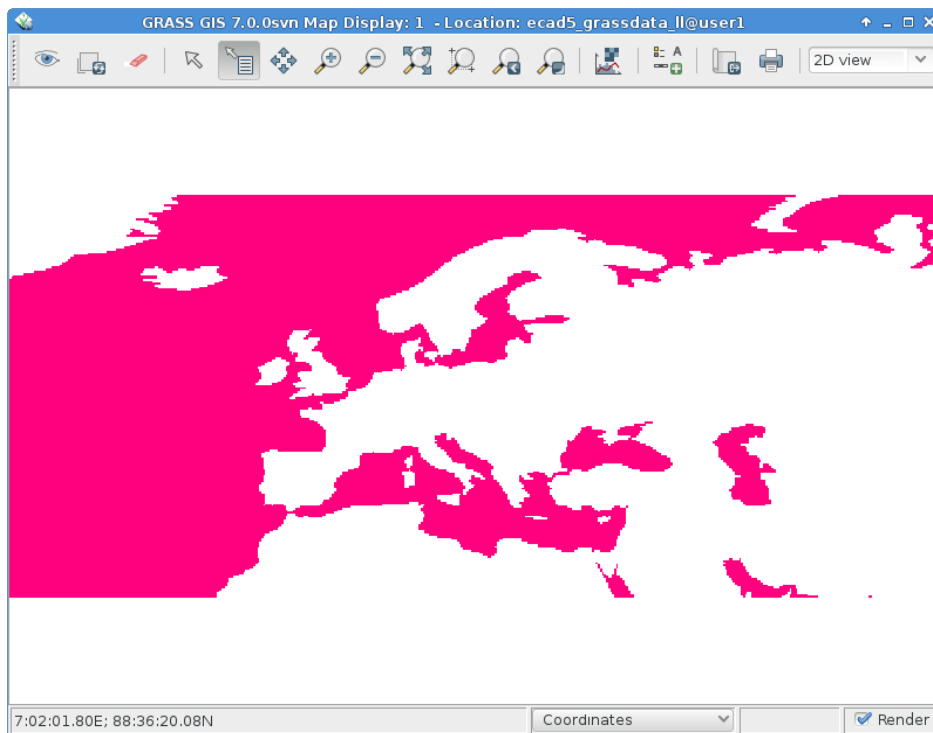
☐ Close dialog on finish

v.to.rast input=country_boundaries@PERMANENT output=country_boundaries use=val

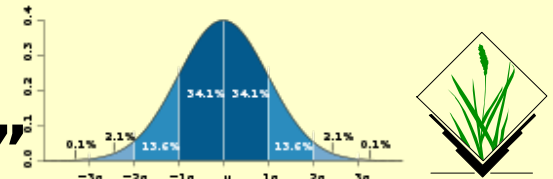
Analysing the relationship between “distance from sea” and “temperature”



```
# invert "land" map to "sea" map to get "distance from sea":  
r.mapcalc "sea = if(isnull(country_boundaries), 1, null() )"  
  
# calculate geodesic distances in [m] (since we are in LatLong)  
# -m      Output distances in [m] instead of map units [deg. here]  
# insert as one line:  
r.grow.distance -m input=sea distance=dist_from_sea \  
                metric=geodesic
```



Analysing the relationship between “distance from sea” and “temperature”



Now we calculate overall average of T_mean

```
# see list of maps and decide the pattern for the names:
g.list raster
```

```
# test the pattern selection
g.list raster pattern="tmean.1981_2010.*.avg"
```

```
# test the pattern selection with separator:
g.list raster pattern="tmean.1981_2010.*.avg" sep=","
```

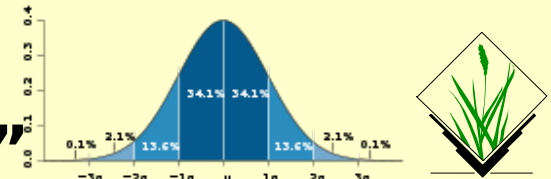
```
# aggregate this time series
g.list raster pattern="tmean.1981_2010.*.avg" \
    output=list_tmean.csv
r.series file=list_tmean.csv \
    output=tmean.1981_2010.avg method=average
```

```
# check extended univariate statistics
r.univar -e tmean.1981_2010.avg
```

```
#... validate: are min/max/mean values reasonable?
```

Cont.

Analysing the relationship between “distance from sea” and “temperature”



```
# random sampling of 1000 "meteo-stations": generate points.
# we sample many due to "stations" falling into the sea = no-data:
v.random output=t_mean_europe_random1000 n=1000

# assign attributes to these random vector points
v.db.addtable t_mean_europe_random1000 \
    columns="t_mean double precision, dist_sea double precision"

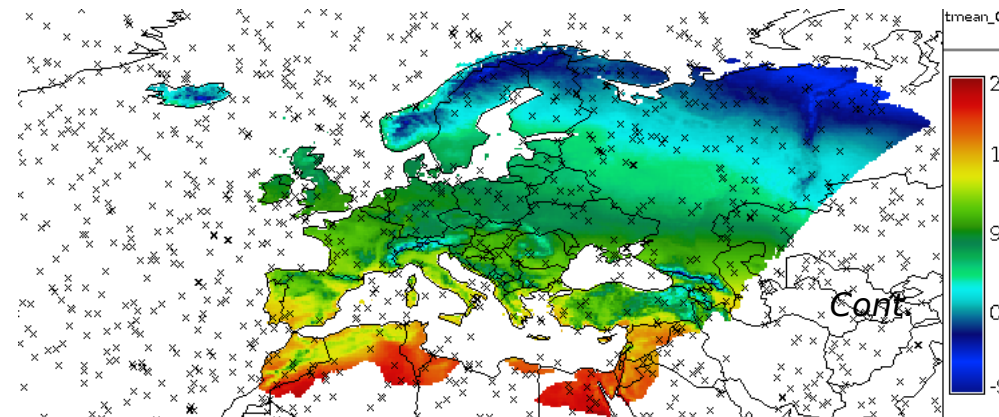
# was the table created?
v.info -c t_mean_europe_random1000
v.db.select t_mean_europe_random1000
```

→ we now have the points with a yet unpopulated attribute table column.

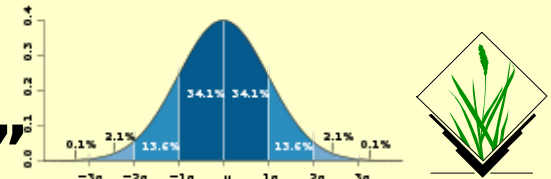
Note: we use v.random here to illustrate the use of SQL – there is also r.random which extracts vector values right away

Now display random points map

note: those virtual “stations” in the sea we'll drop later



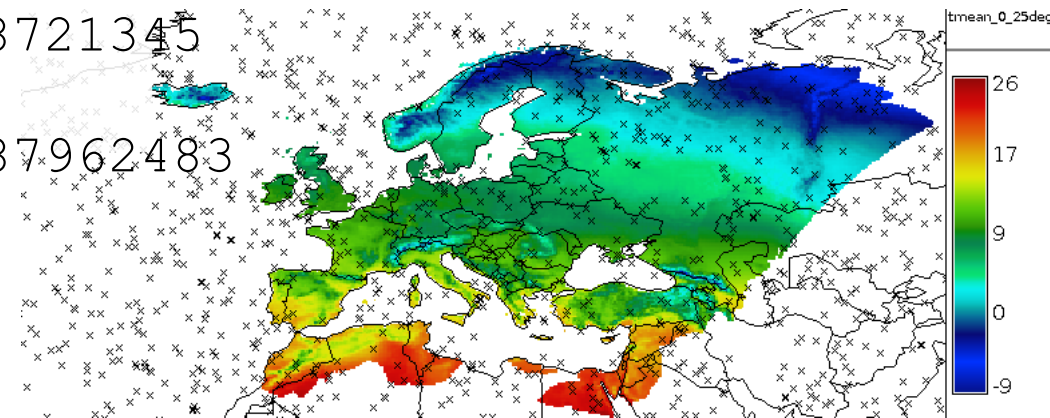
Analysing the relationship between “distance from sea” and “temperature”



Now we populate the attribute table by querying the raster maps at the positions of the random vector points:

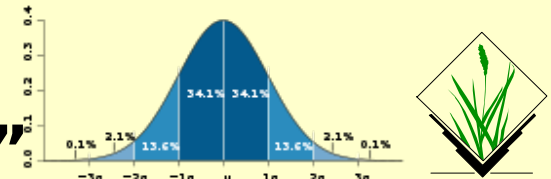
```
# fetch attrib. values from raster maps:
#       distance from sea and t_mean
v.what.rast map=t_mean_europe_random1000 \
          raster=tmean.1981_2010.avg column=t_mean
v.what.rast map=t_mean_europe_random1000 \
          raster=dist_from_sea column=dist_sea

# verify
v.db.select t_mean_europe_random1000 separator=comma
cat,t_mean,dist_sea
1,,0
2,,836479.019992906
3,14.420371055603,179800.318721345
4,,0
5,1.48298296425492,91898.5987962483
...
```



Cont.

Analysing the relationship between “distance from sea” and “temperature”

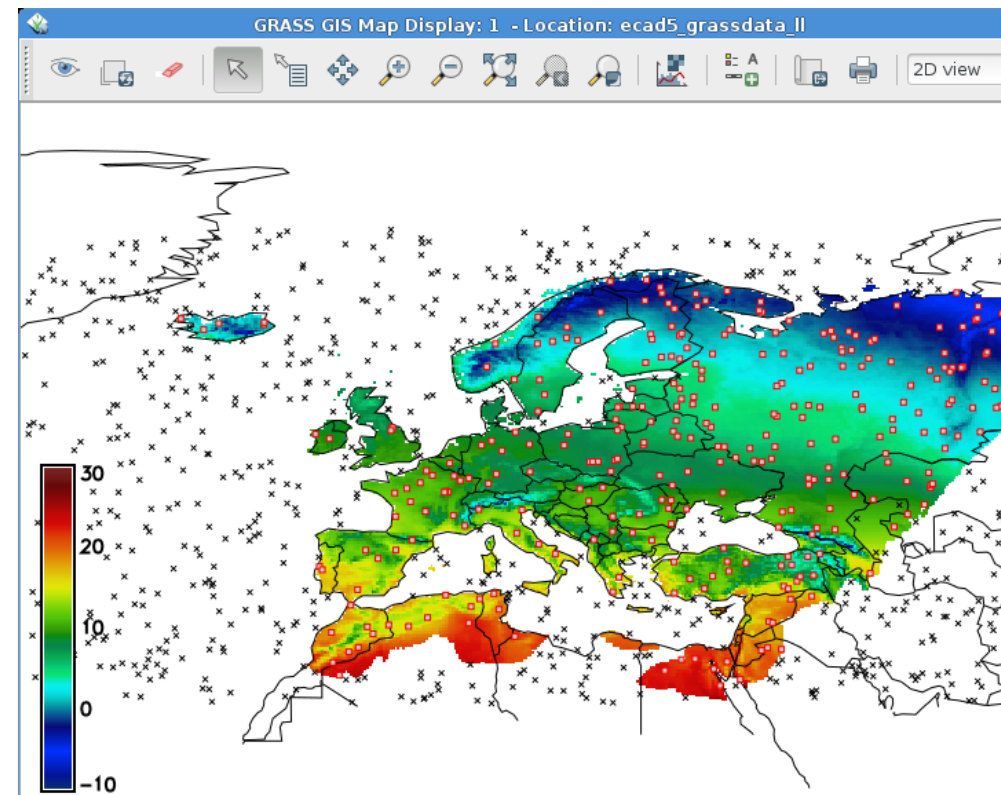


Now we have the prepared vector map with attributes

```
# Now we drop stations positioned in the sea, as well as  
# no data stations using a SQL where statement (approx.  
# 1/3 of the point will remain):
```

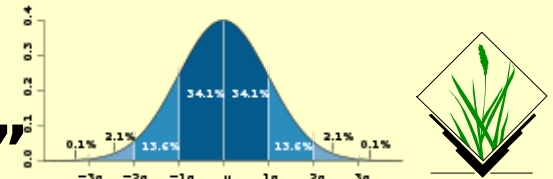
```
v.extract input=t_mean_europe_random1000 \  
  where="dist_sea > 0.0 AND t_mean NOT NULL" \  
  output=t_mean_europe_meteo
```

```
# show 'em ( or use wxGUI)  
d.vect t_mean_europe_meteo \  
  color=red \  
  icon=basic/box  
d.vect country_boundaries \  
  type=boundary
```



Cont.

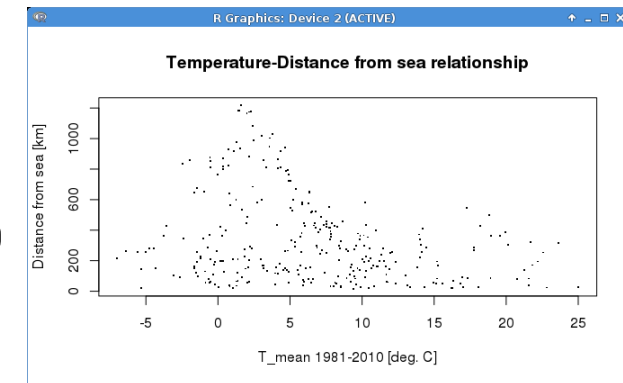
Analysing the relationship between “distance from sea” and “temperature”



```
# Now we analyse these vector point data in R
R
library(rgrass7)
mydata <- readVECT("t_mean_europe_meteo")
summary(mydata)
```

```
# show as points map; histogram plot of T_mean
spplot(mydata)
hist(mydata$t_mean)
```

```
# rescale dist_from_sea to km
mydata$dist_sea <- mydata$dist_sea/1000.0
summary(mydata)
```

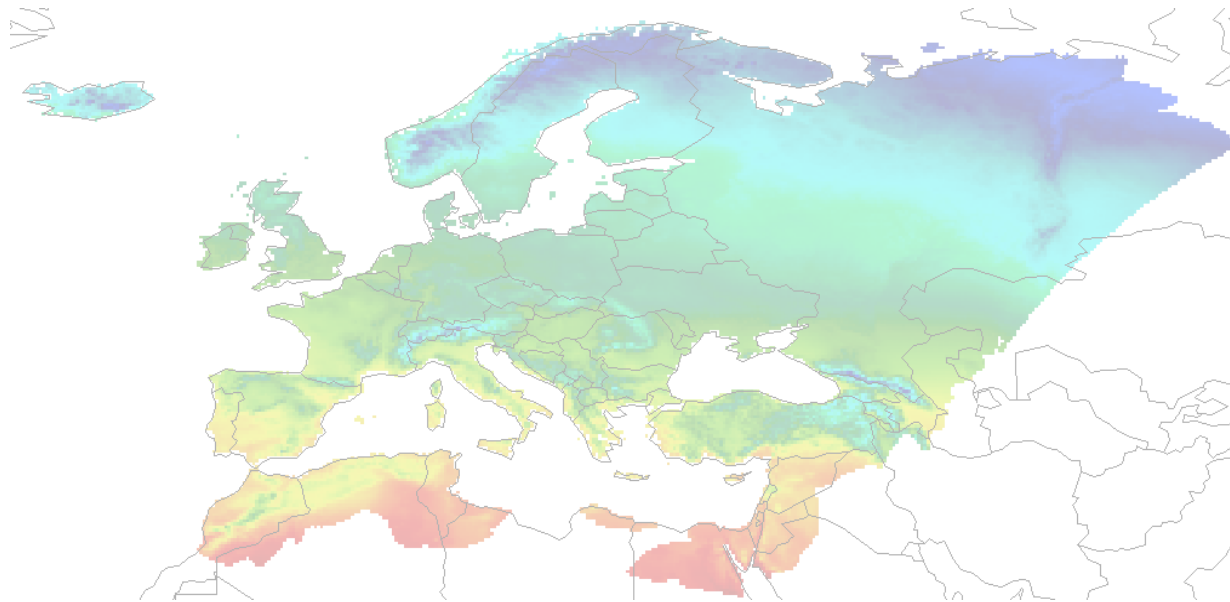


```
# scatterplot
plot (mydata$dist_sea ~ mydata$t_mean, cex=2, pch=".",
      xlab="T_mean 1981-2010 [deg. C]",
      ylab="Distance from sea [km]",
      main="Temperature-Distance from sea relationship")
```

...well, not much to be seen: Across climatic zones we don't find a simple relationship! So we try harder now...



Linear models with GRASS GIS and R



Relationship elevation, latitude, distance from sea and temperature



Next round: we introduce more variables (and `q()` from R for this)

```
r.mapcalc "latitude = y()"
```

```
# add new column and values
```

```
v.db.addcolumn t_mean_europe_meteo \  
  columns="latitude double precision, \  
  elevation double precision"
```

```
v.what.rast map=t_mean_europe_meteo \  
  raster=latitude column=latitude
```

```
v.what.rast map=t_mean_europe_meteo \  
  raster=elev_ecad column=elevation
```

```
# update distance from sea to km in database
```

```
v.db.update t_mean_europe_meteo column=dist_sea \  
  qcolumn="dist_sea / 1000.0"
```

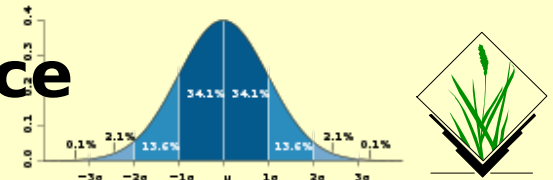
```
# verify
```

```
v.db.select t_mean_europe_meteo  
cat|t_mean|dist_sea|latitude|elevation  
1|3.97897478441397|46.6026245546311|63.125|355.108  
3|10.1735154750446|297.070578809406|39.125|1100.738  
9|4.73268810908|790.035381126579|53.125|219.3317
```

```
...
```

Cont.

Relationship elevation, latitude, distance from sea and temperature



Restart R in your GRASS GIS session

R

```
library(rgrass7)
```

```
mydata <- readVECT("t_mean_europe_meteo")
```

```
summary(mydata)
```

```
# get rid of nodata
```

```
summary(mydata)
```

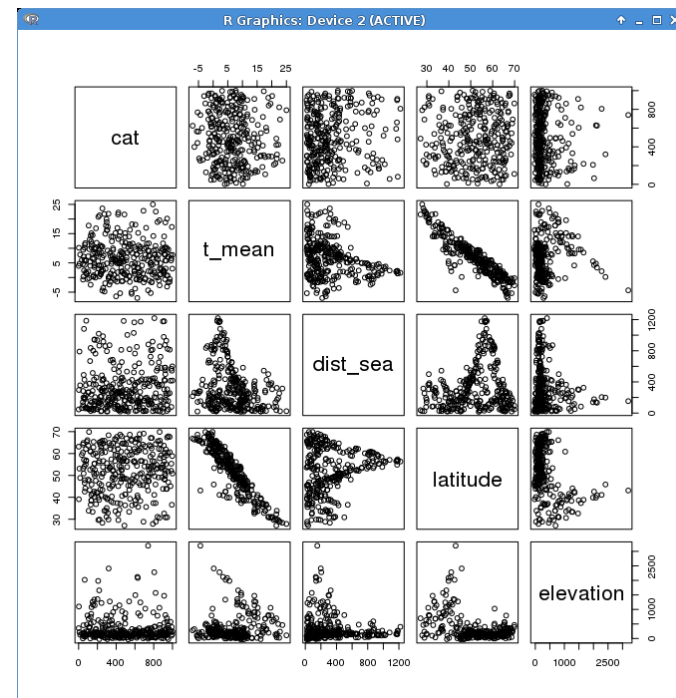
```
mydata <- mydata[!is.na(mydata$t_mean), ]
```

```
# plot the variables
```

```
# against each other
```

```
plot(mydata@data)
```

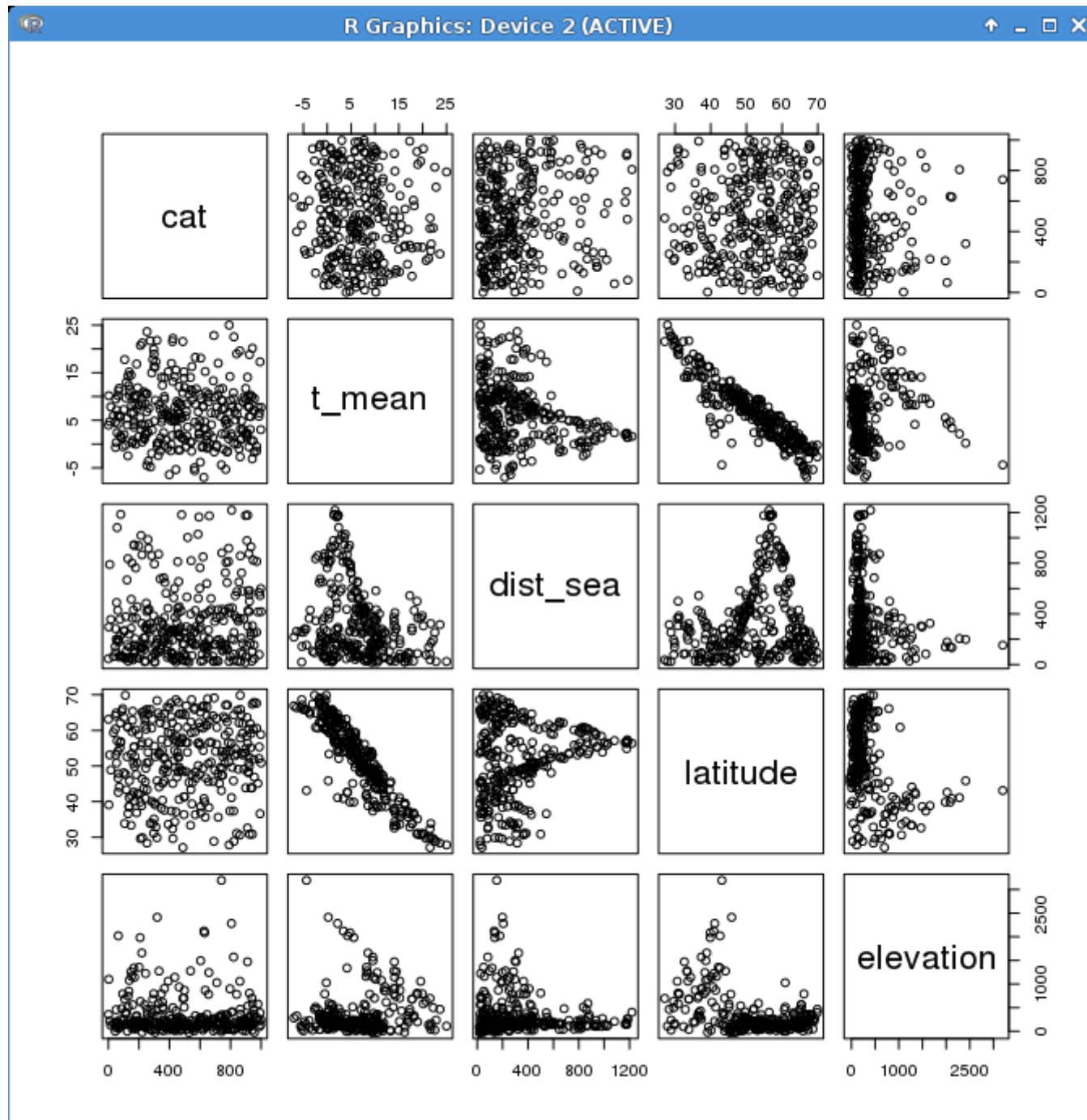
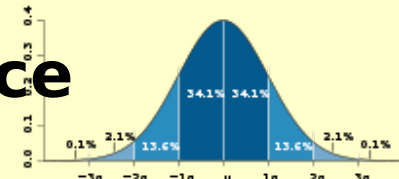
Hints: there is also an extra raster map available: "ecad_solarangle_elev.tif" in order to add yet another Variable, calculated with [r.sunhours](#) in GRASS GIS



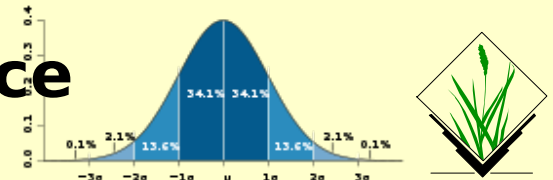
*See next page
for larger figure*

Cont.

Relationship elevation, latitude, distance from sea and temperature



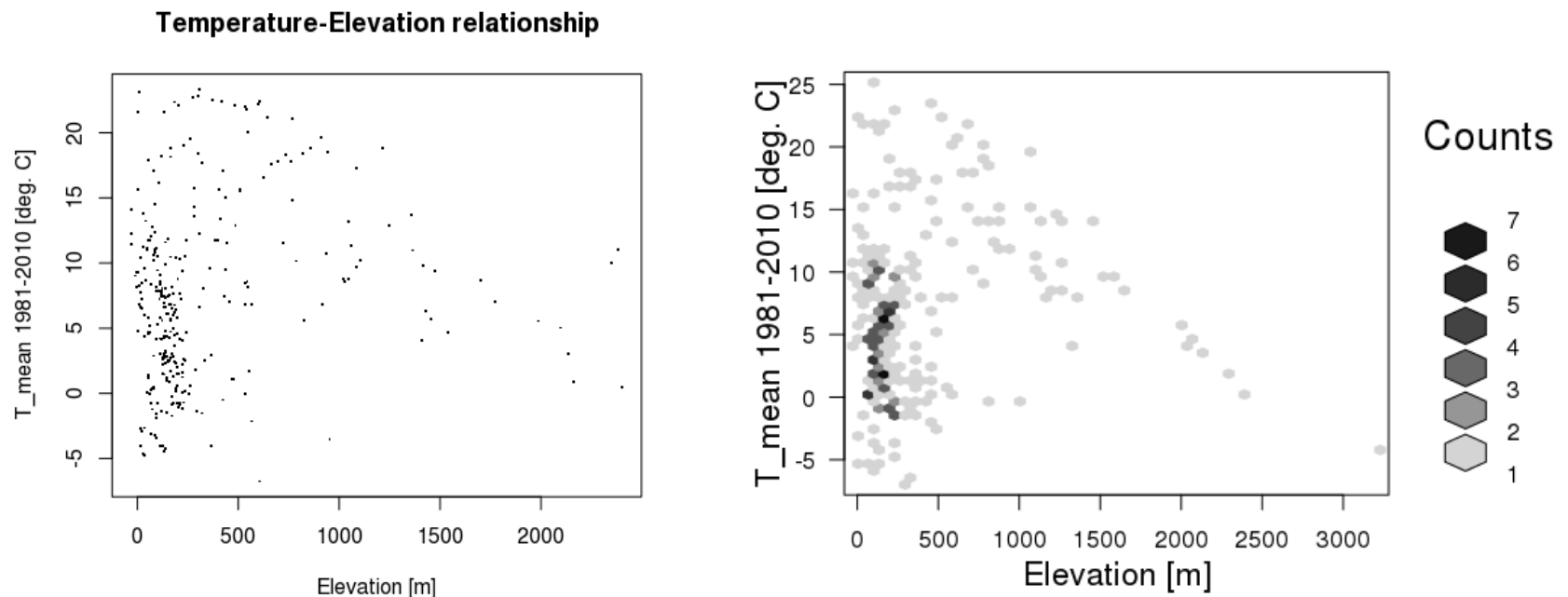
Relationship elevation, latitude, distance from sea and temperature



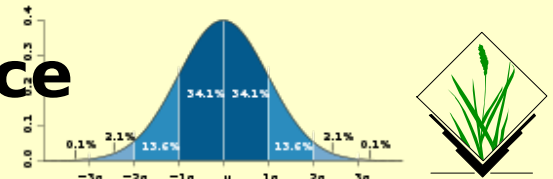
```
# scatterplot T_mean vs Elevation
plot (t_mean ~ elevation, mydata, cex=2, pch=".",
      xlab="Elevation [m]", ylab="T_mean 1981-2010 [deg. C]",
      main="Temperature-Elevation relationship")
```

Hint – Having too many points? Try “hexagon binning”

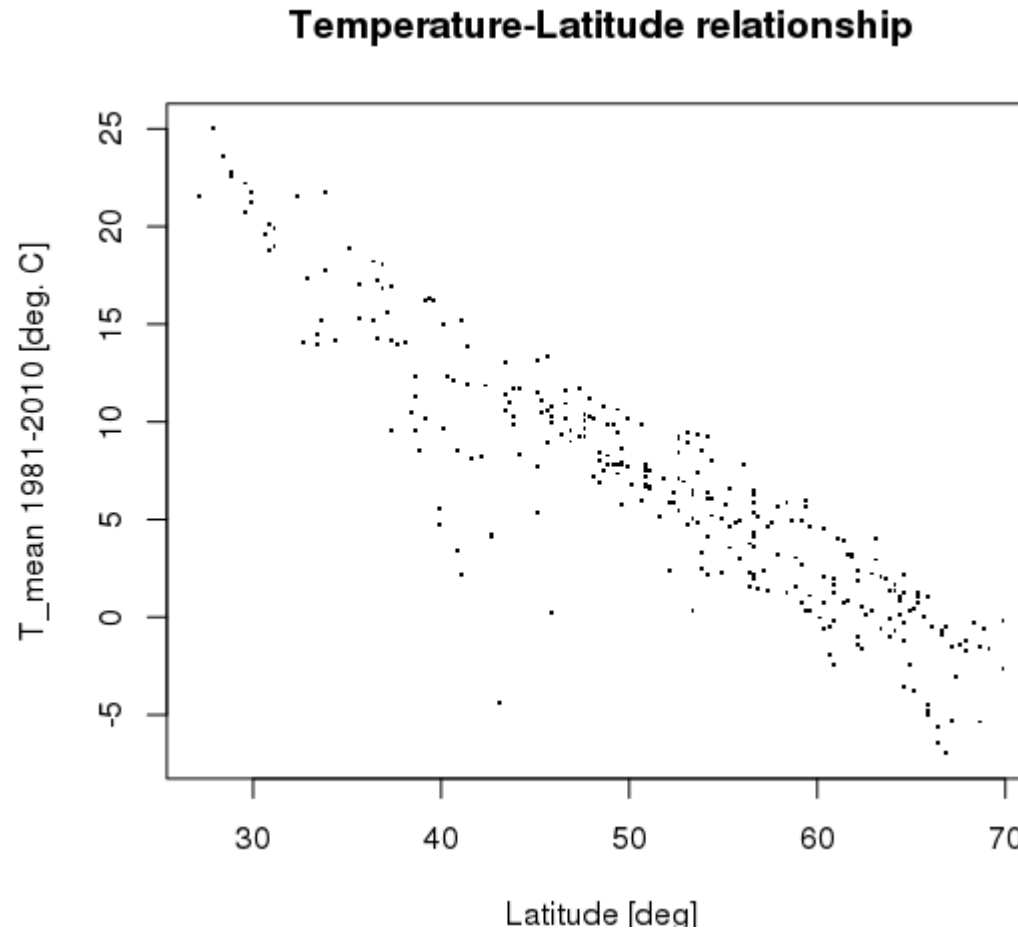
```
# requires: install.packages("hexbin")
library(hexbin)
hbin <- hexbin(mydata$elevation, mydata$t_mean, xbins = 50,
               xlab="Elevation [m]", ylab="T_mean 1981-2010 [deg. C]")
plot(hbin)
```



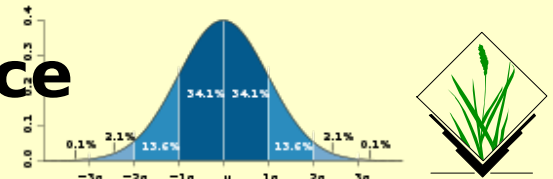
Relationship elevation, latitude, distance from sea and temperature



```
# scatterplot T_mean vs Latitude
plot (t_mean ~ latitude, mydata, cex=2, pch=".",
      xlab="Latitude [deg]", ylab="T_mean 1981-2010 [deg. C]",
      main="Temperature-Latitude relationship")
```



Relationship elevation, latitude, distance from sea and temperature



```
# Linear model T_mean vs Latitude
linmodel <- lm(t_mean ~ latitude, mydata)
abline(linmodel, col="red")
summary(linmodel)
```

Residuals:

*Due to random sampling
Values may slightly differ*

Min	1Q	Median	3Q	Max
-15.7127	-1.1581	0.3328	1.6983	5.2907

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	34.98965	0.74636	46.88	<2e-16 ***
latitude	-0.54793	0.01408	-38.91	<2e-16 ***

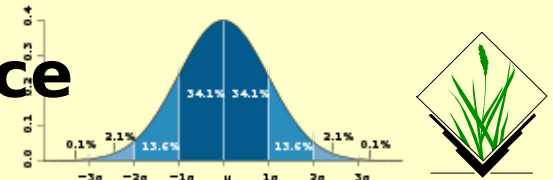
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.602 on 304 degrees of freedom

Multiple R-squared: 0.8328, Adjusted R-squared: 0.8322

F-statistic: 1514 on 1 and 304 DF, p-value: < 2.2e-16

Relationship elevation, latitude, distance from sea and temperature



Perform the same calculation in GRASS GIS

```
# (use CTRL-Z in R to suspend ("park") it,  
# and "fg" command to get back into the R session)
```

```
r.regression.line map1=latitude \  
                    map2=tmean.1981_2010.avg -g
```

```
a=36.007617
```

```
b=-0.565162
```

```
R=-0.930083
```

```
N=30311
```

```
F=194291.514322
```

```
meanX=51.388518
```

```
sdX=10.863195
```

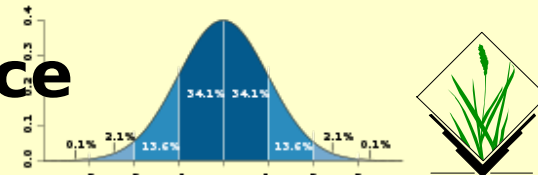
```
meanY=6.964766
```

```
sdY=6.600992
```

BUT:

... note that here ALL raster points are considered here,
not only the random vector points

Relationship elevation, latitude, distance from sea and temperature



Creating a multivariate linear model:

```
#  $y = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_n + e$ 
```

```
#
```

```
# Linear model T_mean vs Elevation + Latitude + dist_from_sea:  
linmodel <- lm(t_mean ~ elevation + latitude + dist_sea, mydata)
```

```
summary(linmodel)
```

Residuals:

Min	1Q	Median	3Q	Max
-4.9705	-0.8431	0.3034	0.9686	3.6243

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	41.3338066	0.5212465	79.30	<2e-16 ***
elevation	-0.0048608	0.0002197	-22.13	<2e-16 ***
latitude	-0.6166385	0.0092093	-66.96	<2e-16 ***
dist_sea	-0.0034204	0.0003131	-10.93	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.532 on 301 degrees of freedom

(1 observation deleted due to missingness)

Multiple R-squared: 0.9422, Adjusted R-squared: 0.9416

F-statistic: 1635 on 3 and 301 DF, p-value: < 2.2e-16

```
coefficients(linmodel)
```

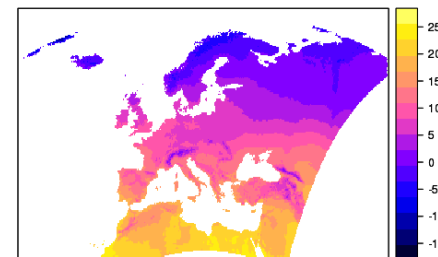
(Intercept)	elevation	latitude	dist_sea
41.333806589	-0.004860809	-0.616638492	-0.003420416

Relationship elevation, latitude, distance from sea and temperature

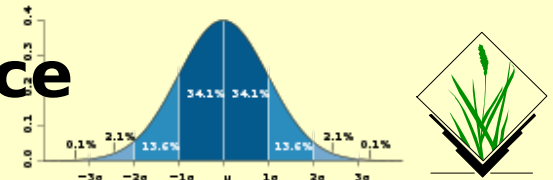


Creating "t_mean_model" the map "the R way..."

```
mymaps <- readRAST(c("elev_ecad", "latitude",  
  "dist_from_sea"))  
summary(mymaps)  
# rename to match names of linear model  
names(mymaps) <- c("elevation", "latitude", "dist_sea")  
summary(mymaps)  
# look at one of the maps  
spplot(mymaps, "elevation")  
  
# convert m to km  
mymaps$dist_sea <- mymaps$dist_sea / 1000.0  
  
# predict T_mean from linear model  
mymaps$t_mean_model <- predict(linmodel, newdata=mymaps)  
summary(mymaps)  
str(mymaps@data)  
spplot(mymaps, "t_mean_model")  
  
# write out to GRASS:  
writeRAST(mymaps, "t_mean_model_R", zcol="t_mean_model")
```



Relationship elevation, latitude, distance from sea and temperature



Creating the "t_mean_model" map the GRASS way...
(interesting approach when dealing with HUGE datasets!)

Linear model

$$y = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_n + \epsilon$$

General formula in GRASS GIS:

```
# r.mapcalc "t_mean_model = $b0 + $b1 * var1 + $b2 * var2
```

we copy-paste the values from the linear model created before
as **shell variables** into the terminal (no white space allowed):

```
# Intercept
```

```
b0=41.3338066
```

```
# elev_ecad
```

```
b1=-0.0048608
```

```
# latitude
```

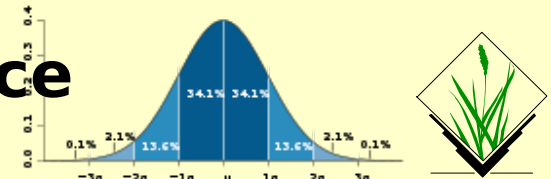
```
b2=-0.6166385
```

```
# dist_from_sea
```

```
b3=-0.0034204
```

Hint: write this into a file and "source" it (. file) into GRASS session

Relationship elevation, latitude, distance from sea and temperature



Creating the map “the GRASS way...” 2/2

```
# we simply use the raster map names + the shell variables
# note: re-adjusting variable "dist_sea" from km to m (= map units)
r.mapcalc "t_mean_model2 = $b0 + $b1 * elev_ecad \
          + $b2 * latitude + $b3/ 1000.0 * dist_from_sea"
```

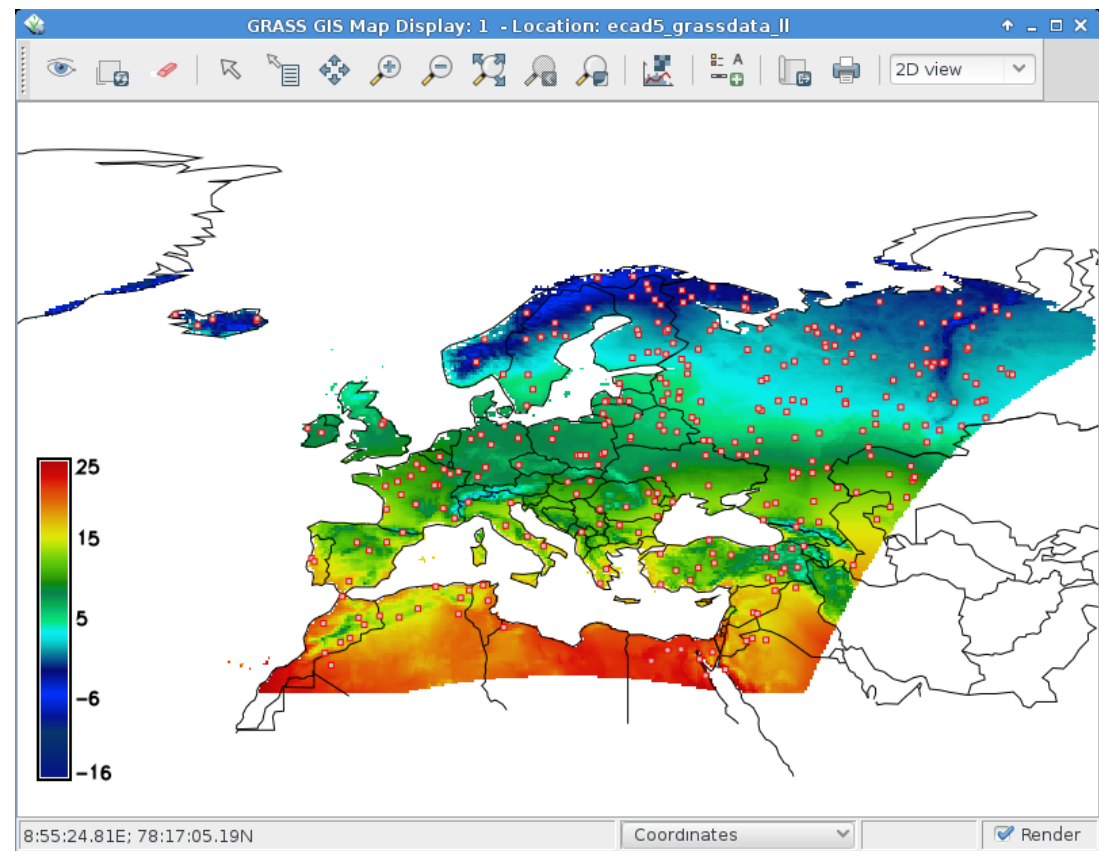
```
r.colors t_mean_model2 color=celsius
```

```
d.mon wx0
```

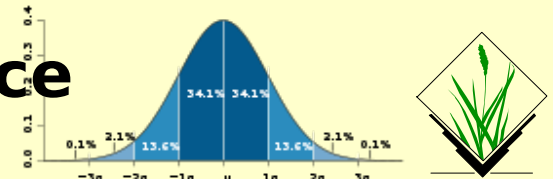
```
d.rast.legend t_mean_model2
```

```
d.vect t_mean_europe_meteo
```

```
d.vect country_boundaries \
    type=boundary
```



Relationship elevation, latitude, distance from sea and temperature

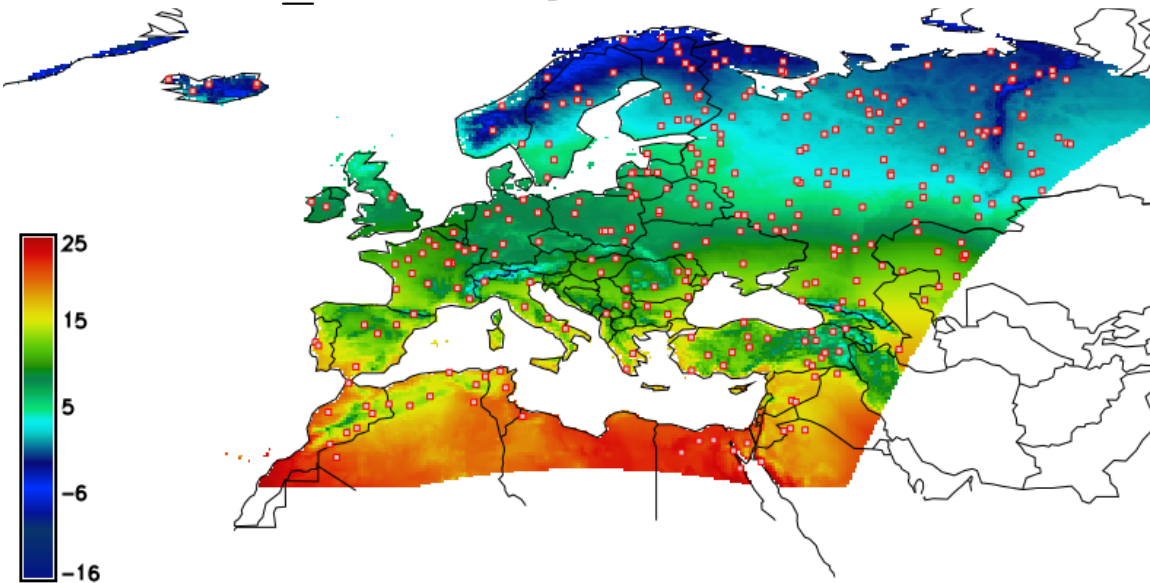


```
# update color table of map to celsius
```

```
r.colors tmean.1981_2010.avg color=celsius
```

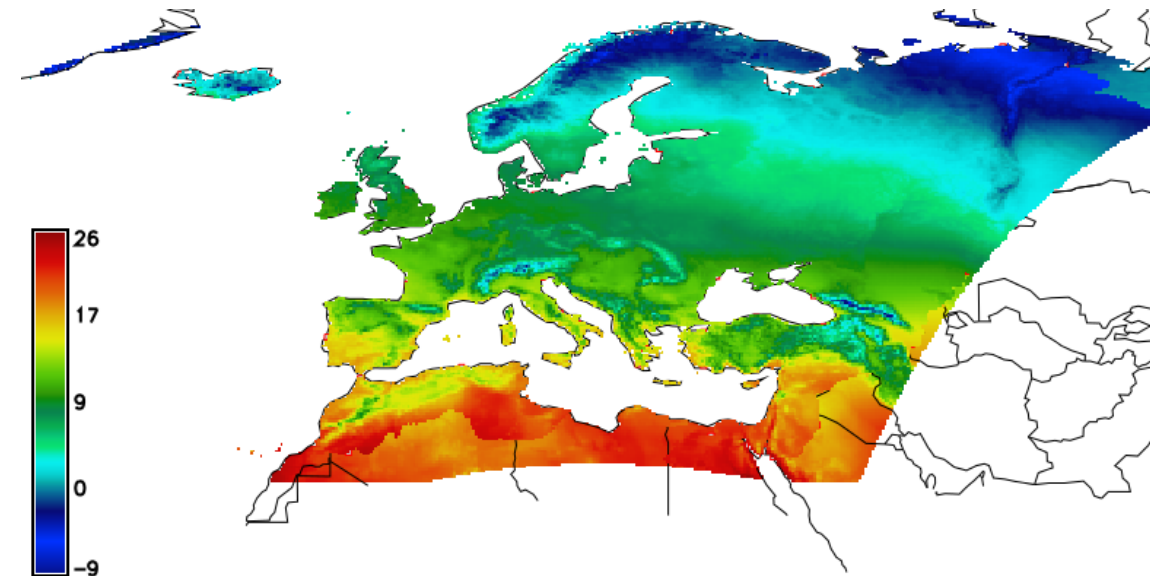
```
d.rast.legend tmean.1981_2010.avg
```

Linear model

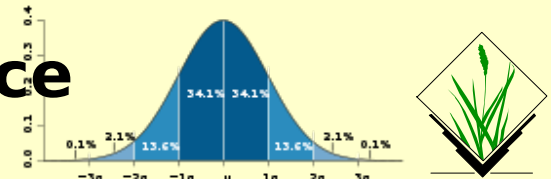


COMPARISON

ECA&D data



Relationship elevation, latitude, distance from sea and temperature



compare simple linear model to real data

```
r.mapcalc "diff_model_ecad = t_mean_model - tmean.1981_2010.avg"
```

```
r.colors diff_model_ecad color=differences
```

deg. C differences map:

```
d.rast.legend diff_model_ecad
```

```
r.univar diff_model_ecad
```

n: 30311

minimum: -5.7443

maximum: 5.54092

range: 11.2852

mean: -0.0370348

mean of absolute values: 1.22533

standard deviation: 1.5751

variance: 2.48093

variation coefficient: -4253.02 %

sum: -1122.56045289236

