

# A roller coaster introduction to spatial analysis using R

Stephanie Kramer-Schadt (IZW)



# The teachers

## Ana Patricia Calderón

Research interests:

- Carnivore/felid conservation
- Connectivity
- Biological corridors

DAAD fellow and Doctoral candidate:

- Connectivity of jaguar populations in Central America
- Universität Potsdam - UFZ – Leibniz IZW; associated with BioMove
- *Aim of project:* improve the connectivity of jaguar populations in Central America to secure their long term viability

# The teachers

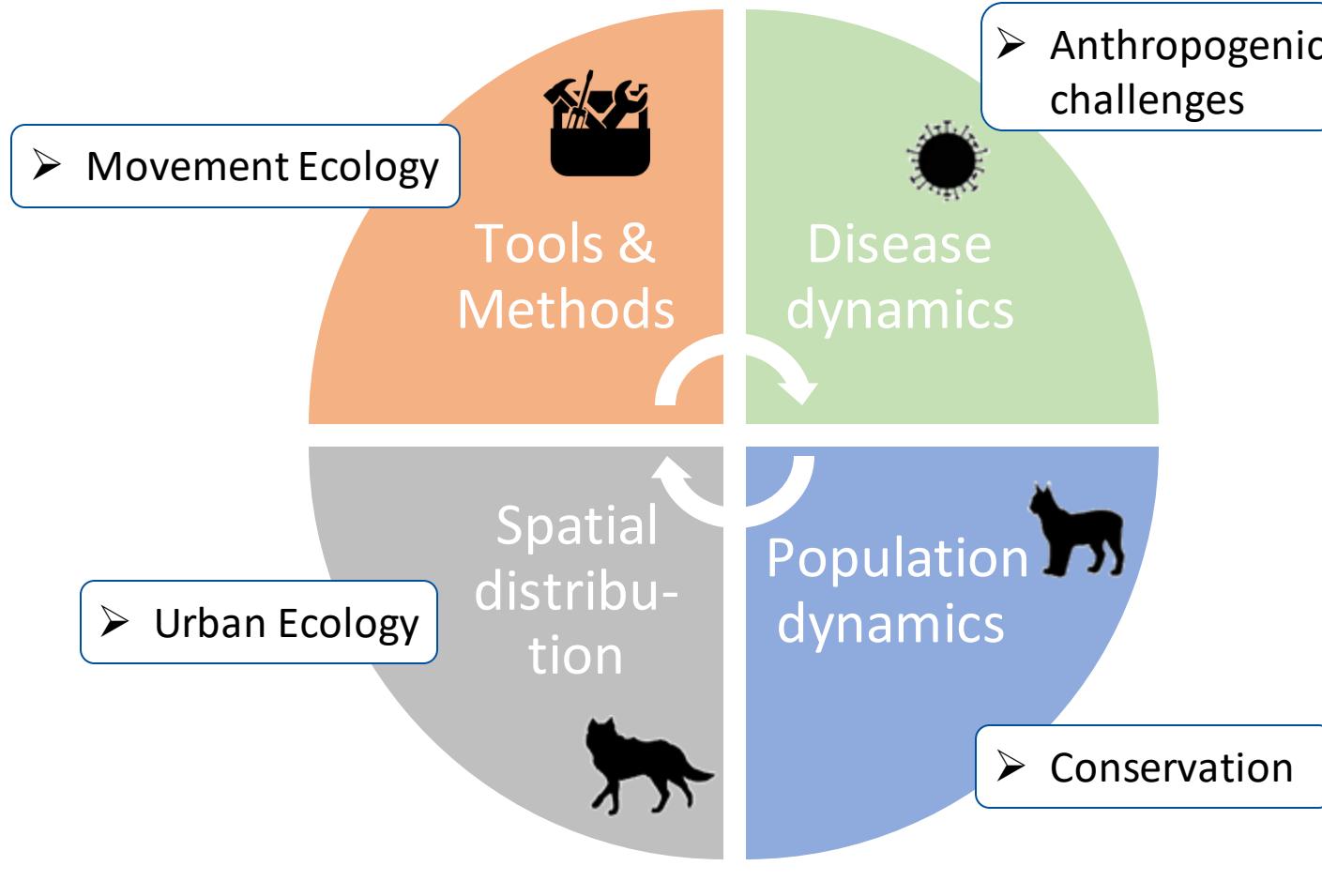
## Cédric Scherer



- PhD in Ecology 2019  
Movement and Disease Ecology  
Simulation Models + Analysis
- PostDoc at IZW since 2019  
Computational Ecology  
Data Analysis + Visualization,  
GIS Data, Lab Support
- Independent DataViz Specialist since  
2020  
Consulting, Design + Coaching

# The teachers

# Stephanie Kramer



# Aim of the Course

To get you going doing spatial operations and analyses in R

- To understand the principle of a GIS (structure, layers,...)
- To see GIS as a tool for analysis
- To be able to visualize and analyze own field data
- To exercise a lot so that loading and manipulating spatial data gets a daily routine

**TAKE HOME MESSAGE:** spatial R is the same as handling basic data types (data.frame, matrix, list)  
– just that you need to understand the added spatial component handling (*geometry*)

# Outline – spatial analysis in R

## DAY 1

### LECTURE (Steph)

- Introduction – What is a GIS?
- Coordinate reference systems, Projections & Analyses
- Data types: vector (shapefiles) and raster data
- Migrating from GUI-GIS to spatial R (basic packages)

### QUESTIONS to pre-course EXERCISES 1 (Patricia)

- First Steps with Loading spatial data / Handling data types in R / Layout

### TIPS'n TRICKS (Steph / Cedric)

- RMarkdown

### EXERCISES 2 (& 3) (Cedric / Steph + Patricia)

- DIY – and use package d6Berlin

Breakout rooms in Zoom  
Work in groups

# Disclaimer

- A GIS does not free you from thinking!
- Definition of the problem --> User!
- The results are only as good as the question, data and analyses! -> this course does not cover statistics!



Leibniz Institute for Zoo  
and Wildlife Research  
IN THE FORSCHUNGSVERBUND BERLIN E.V.

Member of the  
  
Leibniz Association

— Please ask if you do not understand s.th.!

At any time!

So confusing, she couldn't  
explain anything...

Yeah, she spoke way too fast!

That was terrible!

Buh buh



# What is a GIS?

= Geographical Information System

- A system, that is able to display, save, edit, analyze and govern spatial data.
- E.g. used for remote sensing, geography, urban and landscape planning...
- Frequently used systems:  
ESRI ArcGIS<sup>©</sup>,  
GRASS, R, QGIS, SAGA (open source!),  
ERDAS Imagine, ...

# Historic Cartography



Map of Europe of Mercator

Sea travel 15th/16th century

e.g. Gerhard Mercator

Developed a projection in 1569 that is important for sea travel (and aviation) till this day due to its high degree of conformity – the Mercator projection

# First 'Application' (as Analysis Tool)



- 1854 John Snow
- Onset of cholera in London and location of water pumps
  - spatial cluster analysis
  - foundation of epidemiology as a science

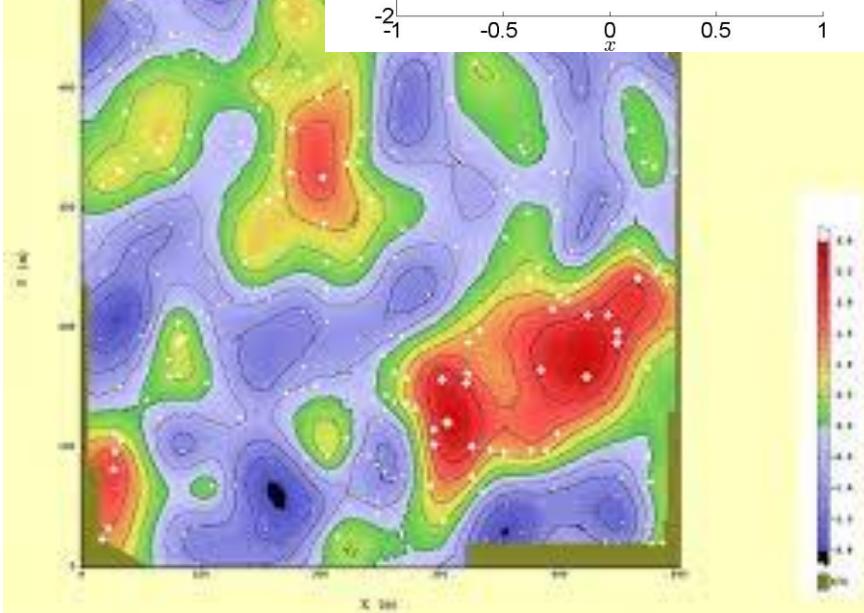
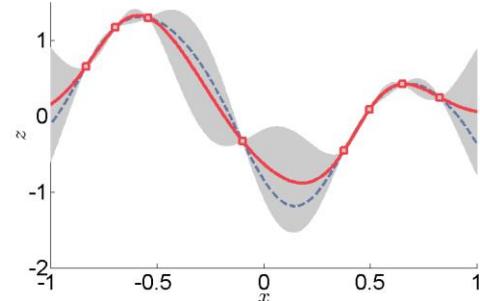
First uses of map-based spatial analysis



# Kriging' – spatial interpolation

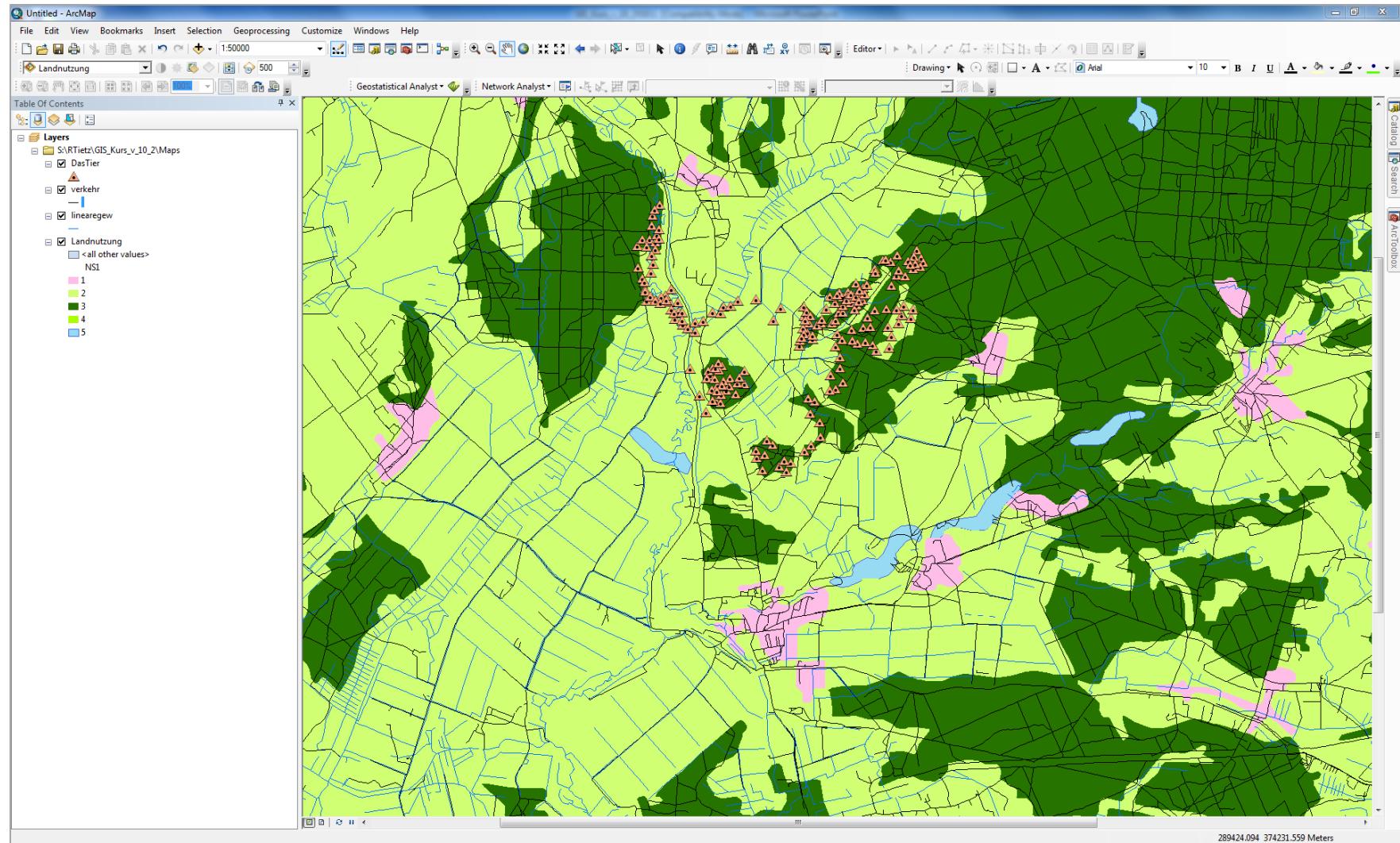


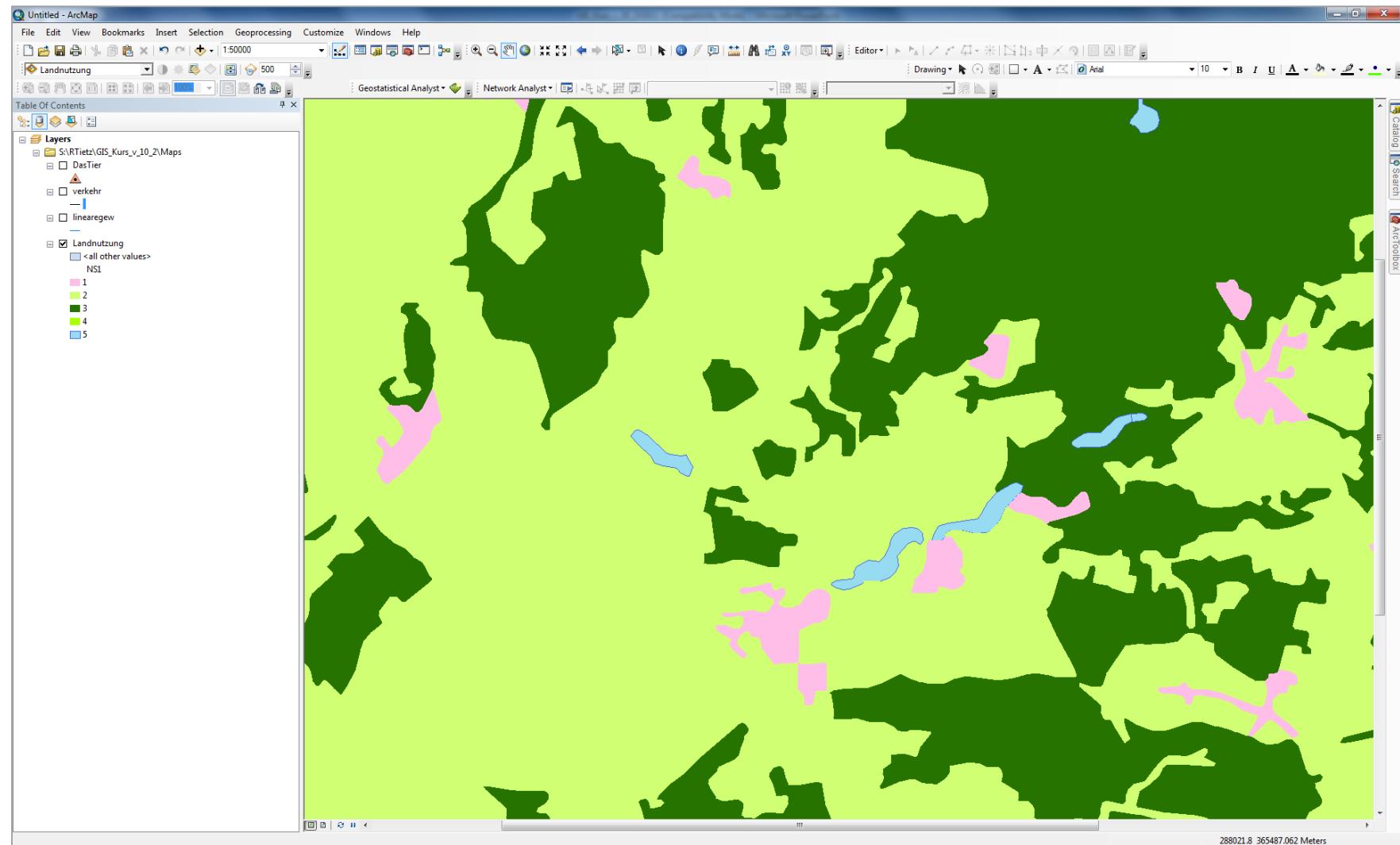
Danie Krige learning the trade, 1939.

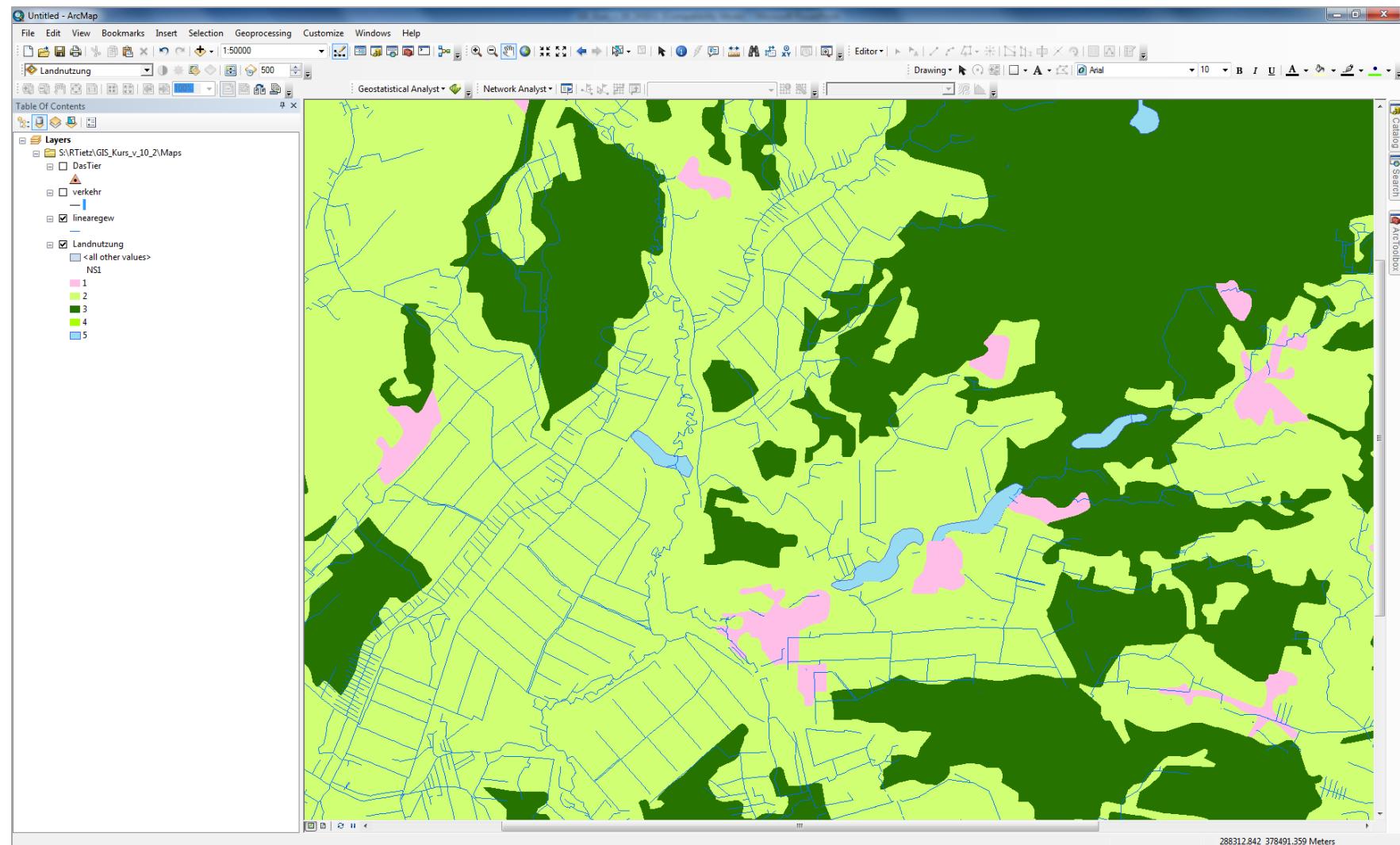


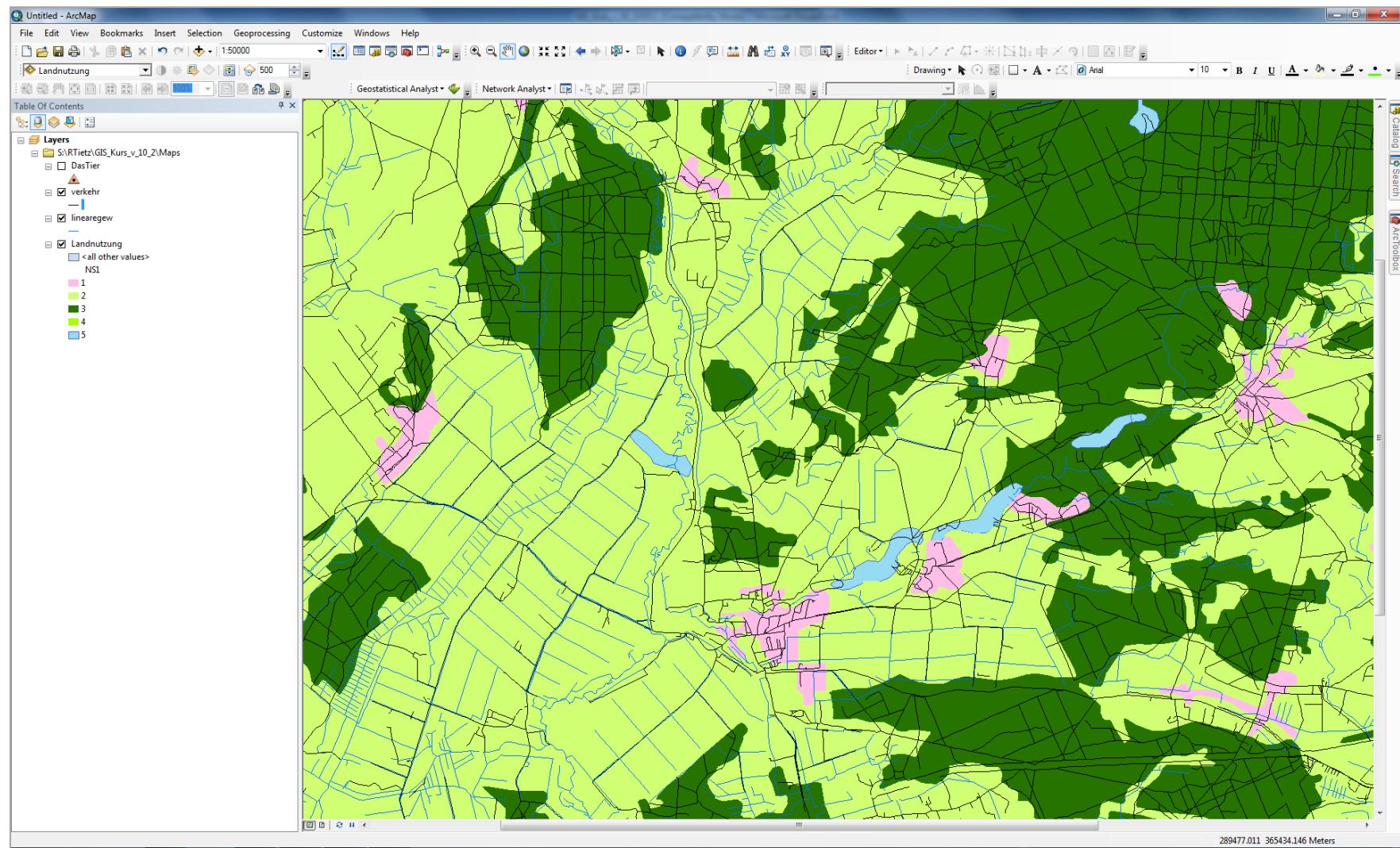
- Daniel G. Krige (\*1919-)
- South African mining engineer
- An interpolation technique
- surrounding measured values are weighted to derive a predicted value for an unmeasured location.
- Weights are based on the distance between the measured points, the prediction locations, and the overall spatial arrangement among the measured points.

# GIS... Just a Digital Map?









# ...or a Database?

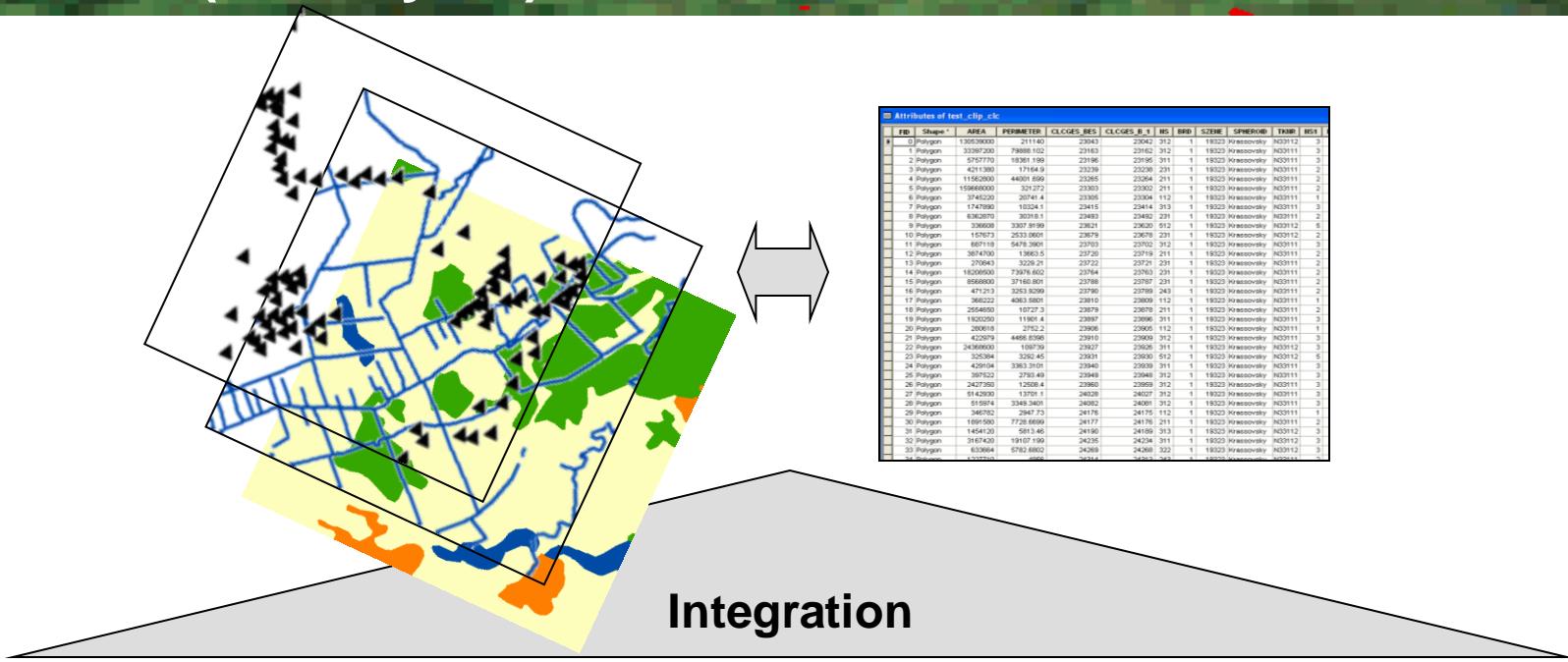
Screenshot of ArcMap showing a map of land use (Ladnutzung) and a corresponding table.

The map displays land use areas colored by category: 1 (light yellow), 2 (pink), 3 (dark green), 4 (yellow-green), and 5 (blue). A network of roads and water bodies is also visible.

The table, titled "Landnutzung", lists 182 selected polygons. The columns represent the following fields:

FID	Shape	AREA	PERIMETER	CLCGES_BES	CLCGES_B_1	NS	BRD	SZENE	SPHEROID	TKNR	NS1	NS2	NS3
0	Polygon	130539000	211140	23043	23042	312	1	19323	Krassovsky	N3311	3	31	312
1	Polygon	3397200	79888.1	23163	23162	312	1	19323	Krassovsky	N3311	3	31	312
2	Polygon	5757770	18361.2	23196	23195	311	1	19323	Krassovsky	N3311	3	31	311
3	Polygon	4211380	17164.9	23239	23238	231	1	19323	Krassovsky	N3311	2	23	231
4	Polygon	11562800	44001.7	23265	23264	211	1	19323	Krassovsky	N3311	2	21	211
5	Polygon	159668000	321272	23303	23302	211	1	19323	Krassovsky	N3311	2	21	211
6	Polygon	3745220	20741.4	23305	23304	112	1	19323	Krassovsky	N3311	1	11	112
7	Polygon	1747890	10324.1	23415	23414	313	1	19323	Krassovsky	N3311	3	31	313
8	Polygon	6362870	30318.1	23493	23492	231	1	19323	Krassovsky	N3311	2	23	231
9	Polygon	3366060	33079.2	23621	23620	512	1	19323	Krassovsky	N3311	5	51	512
10	Polygon	157673	2533.06	23679	23678	231	1	19323	Krassovsky	N3311	2	23	231
11	Polygon	687118	5478.39	23703	23702	312	1	19323	Krassovsky	N3311	3	31	312
12	Polygon	3874700	13663.5	23720	23719	211	1	19323	Krassovsky	N3311	2	21	211
13	Polygon	270843	3229.21	23722	23721	231	1	19323	Krassovsky	N3311	2	23	231
14	Polygon	18208500	73976.6	23764	23763	231	1	19323	Krassovsky	N3311	2	23	231
15	Polygon	8568800	37160.8	23788	23787	231	1	19323	Krassovsky	N3311	2	23	231
16	Polygon	471213	2353.93	23790	23789	243	1	19323	Krassovsky	N3311	2	24	243
17	Polygon	368222	4063.58	23810	23809	112	1	19323	Krassovsky	N3311	1	11	112
18	Polygon	2554480	10737.3	23879	23878	211	1	19323	Krassovsky	N3311	2	21	211
19	Polygon	1920250	11901.4	23897	23896	311	1	19323	Krassovsky	N3311	3	31	311
20	Polygon	280618	2752.2	23906	23905	112	1	19323	Krassovsky	N3311	1	11	112
21	Polygon	422979	4466.84	23910	23909	312	1	19323	Krassovsky	N3311	3	31	312
22	Polygon	24368600	109739	23927	23926	311	1	19323	Krassovsky	N3311	3	31	311
23	Polygon	325384	3292.45	23931	23930	512	1	19323	Krassovsky	N3311	5	51	512
24	Polygon	429104	3363.31	23940	23939	311	1	19323	Krassovsky	N3311	3	31	311
25	Polygon	397522	2793.49	23948	23947	312	1	19323	Krassovsky	N3311	3	31	312
26	Polygon	2427350	12508.4	23960	23959	312	1	19323	Krassovsky	N3311	3	31	312
27	Polygon	5142930	13701.1	24028	24027	312	1	19323	Krassovsky	N3311	3	31	312
28	Polygon	515974	3349.34	24082	24081	312	1	19323	Krassovsky	N3311	3	31	312
29	Polygon	346782	2947.73	24176	24175	112	1	19323	Krassovsky	N3311	1	11	112
30	Polygon	1891580	7726.67	24177	24176	211	1	19323	Krassovsky	N3311	2	21	211
31	Polygon	1454120	5813.46	24190	24189	313	1	19323	Krassovsky	N3311	3	31	313
32	Polygon	3167420	19107.2	24235	24234	311	1	19323	Krassovsky	N3311	3	31	311
33	Polygon	633684	5782.68	24269	24268	322	1	19323	Krassovsky	N3311	3	32	322

# Both! GIS: spatial layers + database (+ analysis)



## Geographic Information

- Points
- Lines
- Polygons
- (Topography 3D)

## Integration

## Factual information eg:

- building types/ tree types
- road types/ river categories
- cities, inhabitant density
- elevation, precipitation

# ...a Georeferenced Database

A GIS consists of four functional components:  
**acquisition, maintenance, analysis, presentation**

**Acquisition** is the digitization of geoobjects, the input of attribute data and the import of external data, e.g. existing maps, databases or graphics.

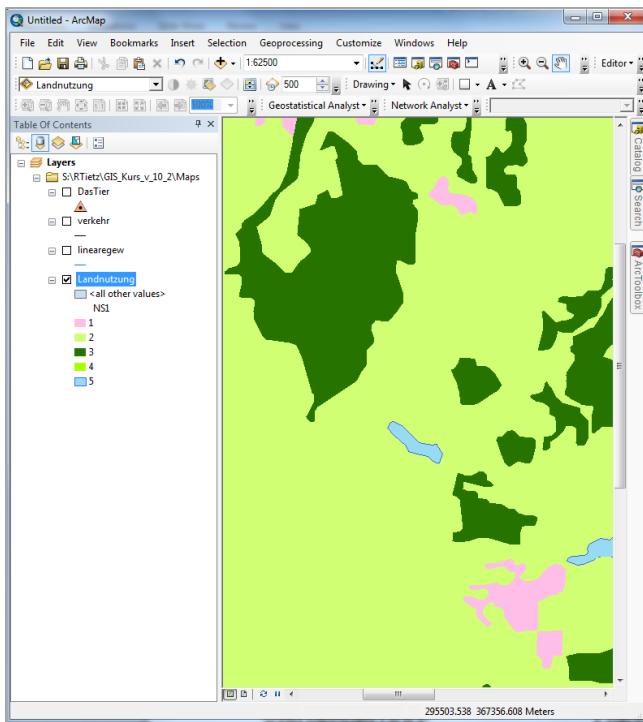
**Maintenance** is the development and organization of the databases for attributes and geometry of the geoobjects.

# ...a Georeferenced Database

The **Analysis** consist of the deduction of new information through logic and geometric links and scanning.

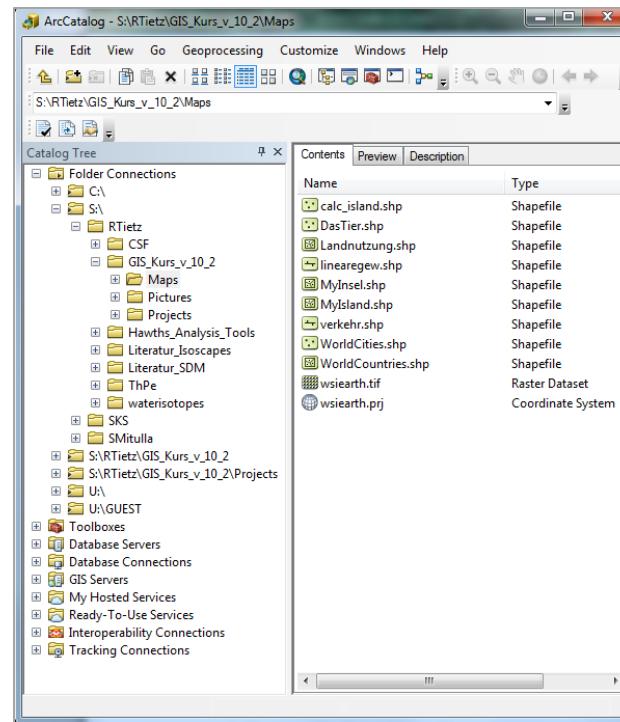
Finally, the **Presentation** displays information in maps and is complemented by texts, diagrams, tables, legends, scales, north arrow etc.

# Program Structure of ArcGIS<sup>®</sup> ESRI



## ArcMap

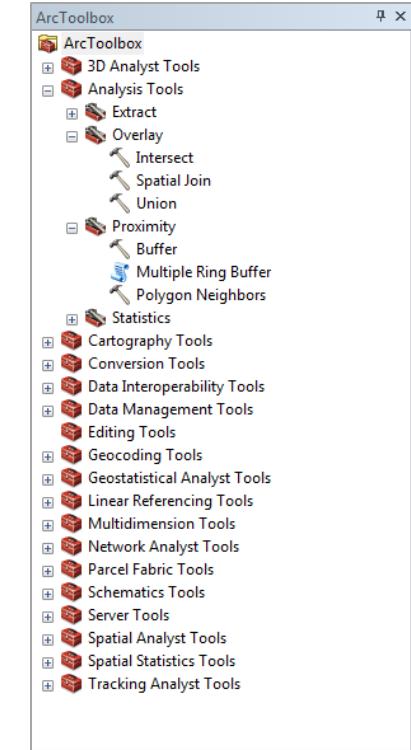
Visualization/  
Presentation



## ArcCatalog

Data Management

Data Aquisition



## ArcToolbox

Analysis

# Programmaufbau QGIS

The screenshot shows the QGIS 2.18.10 interface with the following panels:

- ArcCatalog = Browser Panel:** Located on the left, it displays a tree view of project files and spatial data sources.
- ArcMap = Layer Panel:** Located at the bottom left, it shows a legend for a land use layer named "Landnutzung" with five categories (1-5).
- Map View:** The central area displays a map with various land use categories (green, yellow, pink) and blue line features.
- ArcToolbox = Processing Toolbox:** Located on the right, it lists available processing algorithms categorized by provider.

Toolbar icons are visible along the top edge of the application window.

**Browser-Fenster**

- Home
- Favoriten
- C:/
- D:/
- H:/
- J:/
- M:/
- P:/
- R:/
- DB2
- MSSQL
- Results
- Spatialite
- ArcGisFeatureServer
- ArcGisMapServer
- OWS
- Tile Server (XYZ)
- WCS
- WFS
- WMS

**Layerfenster**

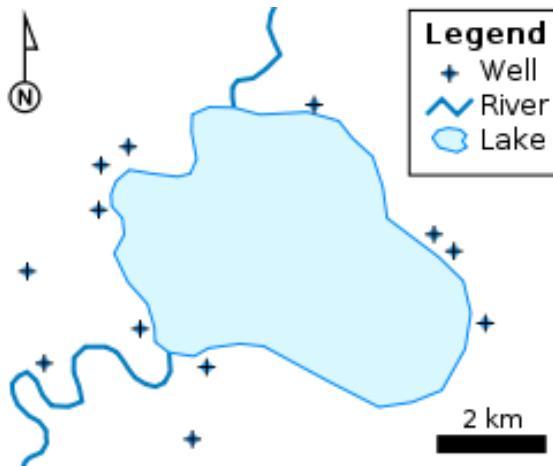
- Landnutzung
  - 1
  - 2
  - 3
  - 4
  - 5

**Verarbeitungswerzeuge**

- Suchen...
- GDAL/OGR [48 Geo-Algorithmen]
- GRASS GIS 7-Befehle [314 Geo-Algorithmen]
- Modelle [0 Geo-Algorithmen]
- QGIS-Geo-Algorithmen [116 Geo-Algorithmen]
- SAGA (2.3.2) [249 Geo-Algorithmen]
- Skripte [0 Geo-Algorithmen]

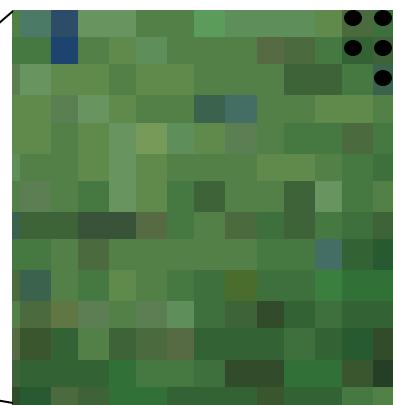
Koordinate 296927,389941 Maßstab 1:166,750 Vergrößerung 100% Drehung 0,0 Zeichnen

# Data types – Points Lines Polygons Raster



## VECTOR DATA (often shapefiles)

- Point: X/Y coordinates
- Line: Length
- Polygon: area/ perimeter  
(closed, ,topology‘)



## RASTER DATA

- ‘Coded Pixels’,  
like points
- GeoTiffs, ascii, grid,  
gpkg...

# ESRI shapefiles

Shapefiles are a data format developed by ESRI used to hold information on spatial objects. They are pretty ubiquitous and can be used by a lot of GIS packages. Shapefiles can hold polygon, line or point data. Despite the name, a shapefile consists of a few different files:

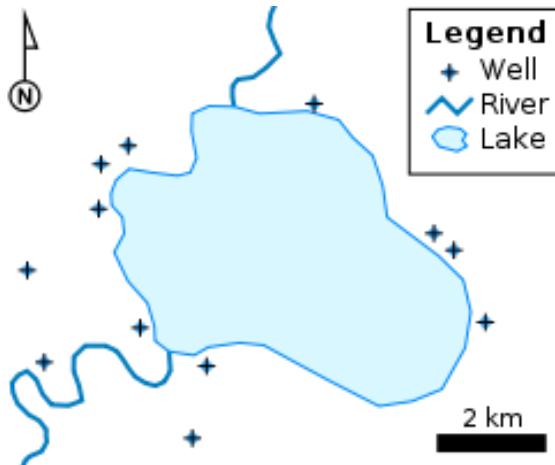
Mandatory files:

- .shp = The main file containing the geometry data
- .shx = An index file
- .dbf = An attribute file holding information on each object

Common additional files:

- .prj = A file containing information on the Coordinate Reference system
- .shp.xml = a file containing object metadata, citations for data, etc.

# Vector Data format: Sbn,... shx, shp



ID	X	Y
1	2	3
1	3	5
1	4	4
1	3	2
1	2	3
2	7	1
2	8	2
2	10	1
2	7	1

## VECTOR DATA

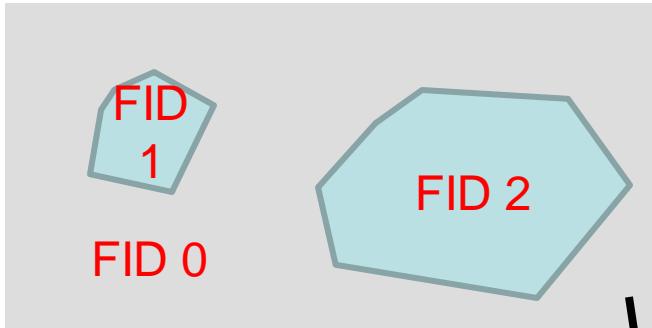
- Point: X/Y coordinates
  - Line: Length
  - Polygon: area/ perimeter  
**(closed, ,topology')**
- > single vector elements are called *features*

Attributes of test_clip_clc								
FID	Shape *	AREA	PERIMETER	CLCGES_BES	CLCGES_B_1	IIS	BRD	S
0	Polygon	130539000	211140	23043	23042	312	1	
1	Polygon	33397200	79888.102	23163	23162	312	1	
2	Polygon	5757770	18361.199	23196	23195	311	1	
3	Polygon	4211380	17164.9	23239	23238	231	1	
4	Polygon	11562800	44001.699	23265	23264	211	1	

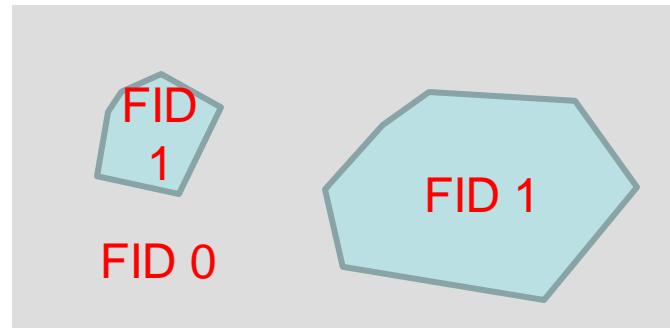
= feature: FID stands for feature-identity

File containing spatial geometry

# Vector Data format: Sbn,... shx, shp



'Union' creates  
a multipolygon



e.g. a shapefile with lakes

ID	X	Y
1	2	3
1	3	5
1	4	4
1	3	2
1	2	3
2	7	1
2	8	2
2	10	1
2	7	1

Never change the FID by hand!

Attributes of test_clip_clc									
FID	Shape *	AREA	PERIMETER	CLCGES_BES	CLCGES_B_1	HS	BRD	S	NAME
0	Polygon	130539000	211140	23043	23042	312	1		Lake 1
1	Polygon	33397200	79888.102	23163	23162	312	1		Lake 2
2	Polygon	5757770	18361.199	23196	23195	311	1		Lake 3
3	Polygon	4211380	17164.9	23239	23238	231	1		Lake 4
4	Polygon	11562800	44001.699	23265	23264	211	1		Lake 5

# In R: *simple feature* vs *spatial polygon df*

```
>  
> test2_sf  
simple feature collection with 158 features and 7 fields  
geometry type:  MULTIPOLYGON  
dimension:      XY  
bbox:           xmin: 108.5986 ymin: -4.943054 xmax: 119.2697 ymax: 7.380556  
epsg (SRID):   NA  
proj4string:    NA  
First 10 features:  
  ID_0 ISO  NAME_0 NAME_1     NAME_2 Shape_Leng Shape_Area  
0  158 MYS  Malaysia Sabah Lahad Datu  6.7475367  0.62106012 MULTIPOLYGON (((118.2299 4....  
1  158 MYS  Malaysia Sabah      Papar  1.6727672  0.10065591 MULTIPOLYGON (((116.0144 5....  
2  158 MYS  Malaysia Sabah  Penampang  0.9548857  0.03951545 MULTIPOLYGON (((116.0477 5....  
3  158 MYS  Malaysia Sabah Pensiangan 4.0257859  0.49725547 MULTIPOLYGON (((116.5357 5....  
4  158 MYS  Malaysia Sabah      Pitas  2.9223456  0.11955352 MULTIPOLYGON (((117.2762 6....  
5  158 MYS  Malaysia Sabah      Ranau  2.4815704  0.24027591 MULTIPOLYGON (((116.9182 6....  
6  158 MYS  Malaysia Sabah Sandakan  7.1555231  0.18354676 MULTIPOLYGON (((117.9468 5....  
7  158 MYS  Malaysia Sabah Semporna  6.0198462  0.09494946 MULTIPOLYGON (((118.6294 4....  
8  158 MYS  Malaysia Sabah Sipitang  2.9190984  0.22604467 MULTIPOLYGON (((115.7132 5....  
9  158 MYS  Malaysia Sabah Tambunan 1.5307563  0.12152936 MULTIPOLYGON (((116.6079 5....
```

Closer to ESRI  
shapefile  
attribute table

{Sf}

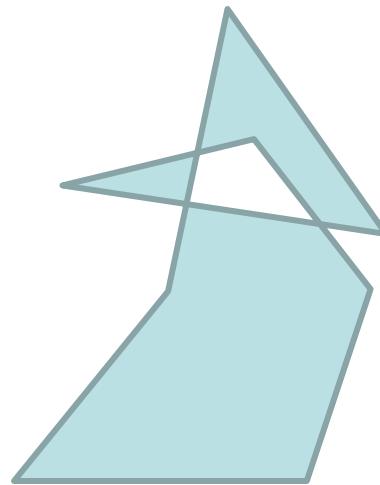
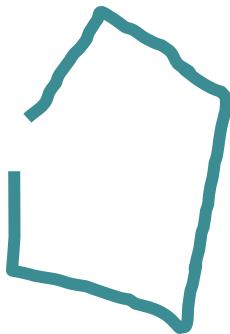
= saved in the geometry as list (POLYGON)  
Or as list in a list (MULTIPOLYGON)

```
class       : SpatialPolygonsDataFrame  
Features    : 158  
extent      : 108.5986, 119.2697, -4.943054, 7.380556  (xmin, xmax)  
coord. ref. : NA  
variables   : 7  
names       : ID_0, ISO,  NAME_0, NAME_1, NAME_2, shape_Leng, sh  
min values  :  34, BRN,  Brunei, Belait,   Amo,  0.2036641, 0.0  
max values  : 158, MYS, Malaysia, Tutong,  ukong, 30.6175230, 3.1  
> |
```

{sp}

# Problems with the topology

- Geometry not closed or not clean



Handling geometry issues:

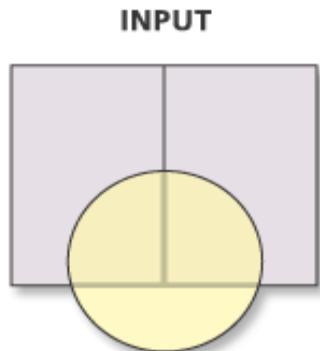
package ***sf*** ***several functions***, e.g., `st_make_valid()`

<https://r-spatial.org/r/2017/03/19/invalid.html>

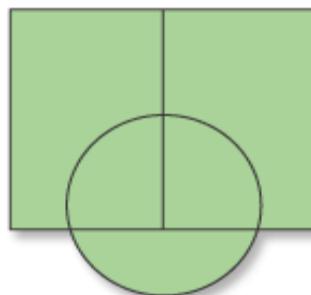
package ***cleangeo***

<https://cran.r-project.org/web/packages/cleangeo/vignettes/quickstart.html>

# Toolbox: Example of analyses with polygons (shapefiles)

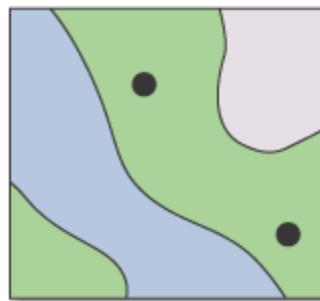


**OUTPUT**

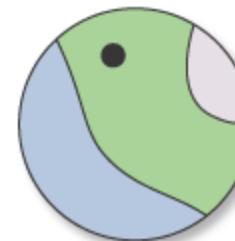
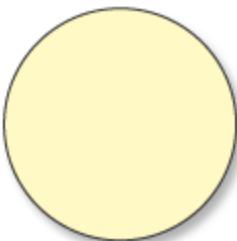


**UNION**

In R: gUnion()  
st\_union()

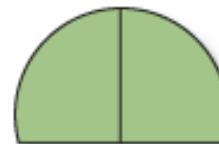
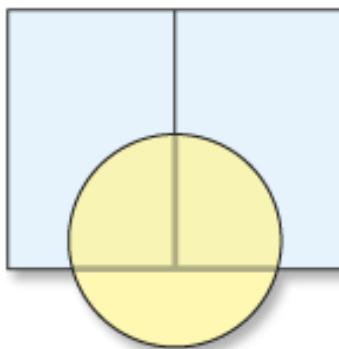


+



**CLIP**

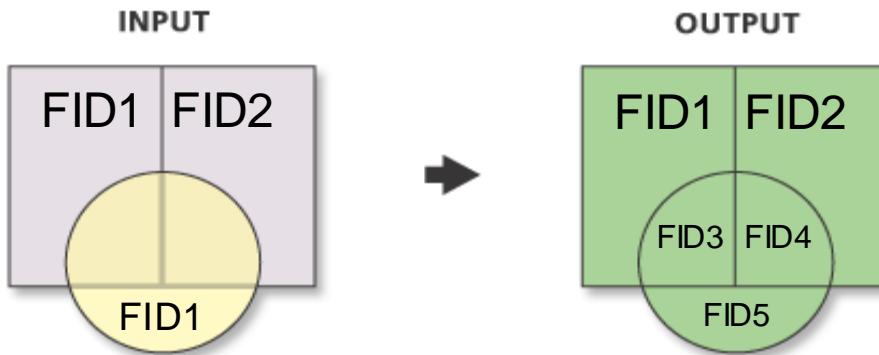
In R: crop()  
extract()  
gIntersection()  
st\_intersection()



**INTERSECT**

In R:  
gIntersection()  
st\_intersection()

# OBS!

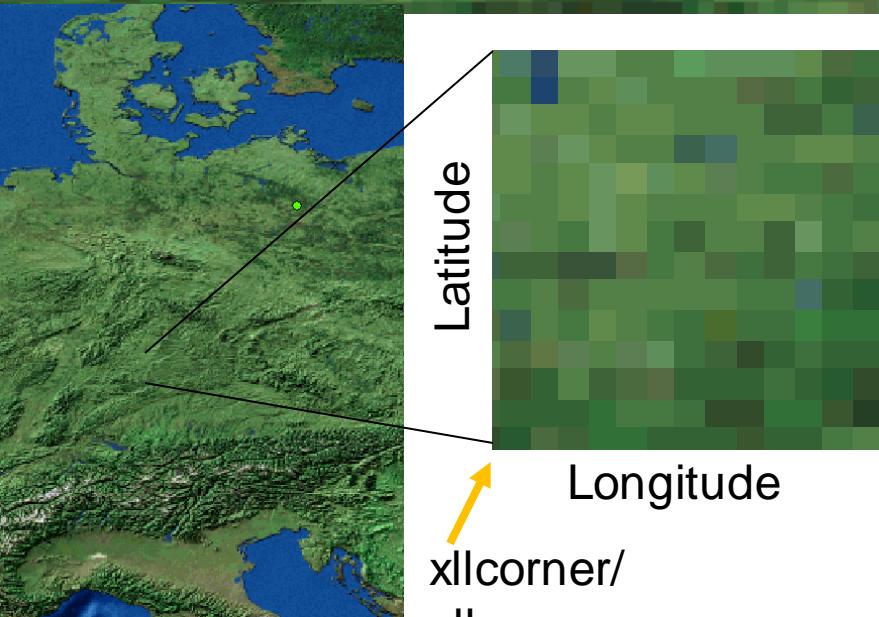


After spatial operation, the FID changes! And the geometry!

And with that also the spatial calculations!

Remember to update/ recalculate the column with  
the area size/ perimeter/ distances after spatial operations!

# Raster format – fast and efficient (Modeling)



- Best for continuous data
  - Ascii or GeoTIFF as exchange format

bio\_asc\_22.asc - Editor

Datei Bearbeiten Format Ansicht ?

```
ncols          1423
nrows          1425
xllcorner      108.33412288693
yllcorner      -4.3740129986359
cellsize        0.0083333338
NODATA_value   -9999

0 0 0 0 400 5600 400 0 0 0 1600 0 0 0 0 1600 4400
6400 1200 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1200 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1600 2800 1600 0 0 0 0 800 0 0 4400 800 0 2000 400
2800 400 3200 2400 3200 4400 1600 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 4800 6000 800 0 1600 2000 800 3600 400 0 0
400 4000 2000 0 0 0 0 0 400 5600 1600 0 0 0 0
6400 0 5600 7200 0 0 0 0 0 0 0 0 0 0 0 0 0 3600 8400
6800 0 0 0 0 0 400 4000 6000 6800 800 3600 0 0 0 0
0 0 1600 400 0 0 1200 6800 9600 10000 10000 10000
10000 10000 9600 3600 4000 9600 1200 0 400 2400
10000 9600 4400 1200 400 1600 800 0 0 0 0 0 0 0 0
0 0 0 0 2000 7600 10000 10000 9600 9600 8000
7600 9200 8000 10000 10000 10000 10000 10000 10000
9200 3200 4400 2000 3200 9600 2400 0 4000 9200
5600 3200 0 0 0 0 0 0 0 0 -9999 -9999 -9999 -9999 -
9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -
9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -
```

# In R – raster (getting outdated) and terra

- Bor\_mat: Borneo mean annual temperature raster

```
Bor_mat <- raster(paste(raster_wd, 'Borneo_MAT.asc', sep='/'))  
Bor_mat
```

```
## class      : RasterLayer (raster)      /   SpatRaster (terra)  
## dimensions : 1425, 1423, 2027775 (nrow, ncol, ncell)  
## resolution : 0.008333334, 0.008333334 (x, y)  
## extent     : 108.3341, 120.1925, -4.374013, 7.500988 (xmin, xmax, ymin, ymax)  
## coord. ref. : NA  
## data source : C:/Users/kramer/Dropbox/_GeoData_Mix/Geodata/rasterfiles/Borneo_MAT.  
asc  
## names      : Borneo_MAT
```

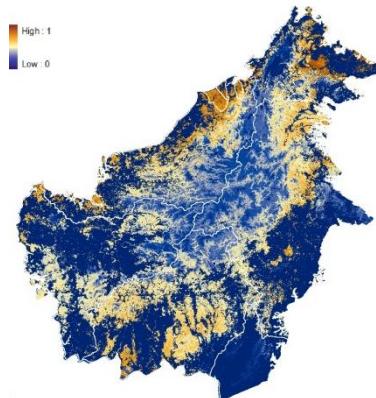
S4-format in raster: slots are accessible via @, i.e.

```
Bor_mat@extent
```

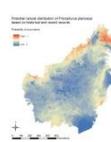
```
## class      : Extent  
## xmin      : 108.3341  
## xmax      : 120.1925  
## ymin      : -4.374013  
## ymax      : 7.500988
```

# Spatial analyses with rasters

- Manipulation (eg reclassification)
- Algebra (eg calculations based on several maps)



$$HSI = (M * L * H)^{1/3}$$

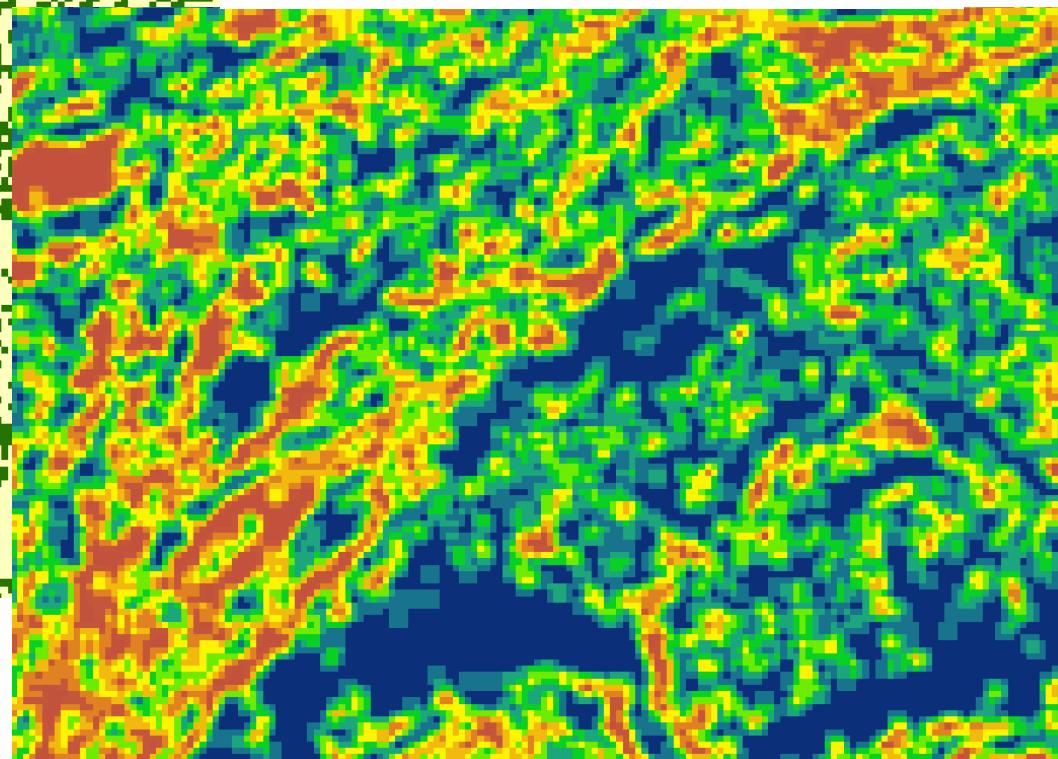


$$= \left( \begin{array}{|c|c|c|} \hline 7 & 10 & 3 \\ \hline 11 & 11 & 8 \\ \hline 8 & 10 & 6 \\ \hline \end{array} \right) * \left( \begin{array}{|c|c|c|} \hline 5 & 7 & 2 \\ \hline 9 & 10 & 4 \\ \hline 8 & 8 & 3 \\ \hline \end{array} \right) * \left( \begin{array}{|c|c|c|} \hline 2 & 3 & 1 \\ \hline 2 & 1 & 4 \\ \hline 0 & 2 & 3 \\ \hline \end{array} \right) )^{1/3}$$

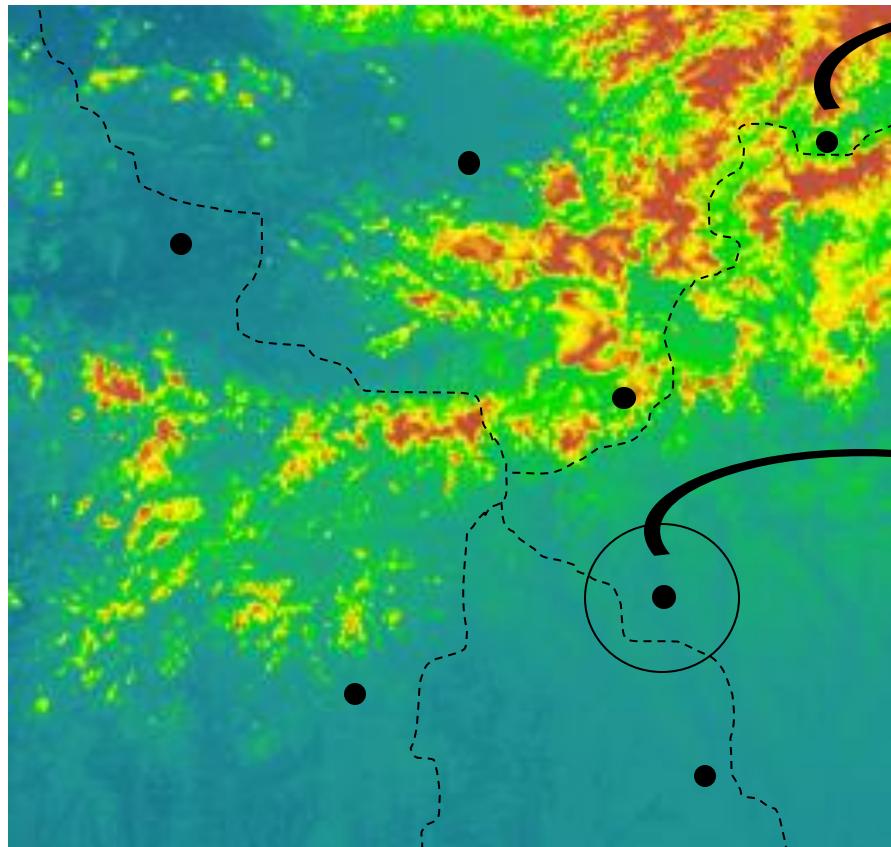
- Modelling (NDVI, ruggedness, neighborhood analyses, cost path, watershed,...)

# Spatial analyses with rasters II

e.g. Connectivity indices  
(Focal Statistics):  
To describe single point  
values across larger area



# Analysis: Building Databases With Spatial Info



## Per Point (Extraction):

- Map value  
(Land use, climate)
- area/ perimeter of patch
- distances to roads...

In R:  
extract  
over  
st\_join

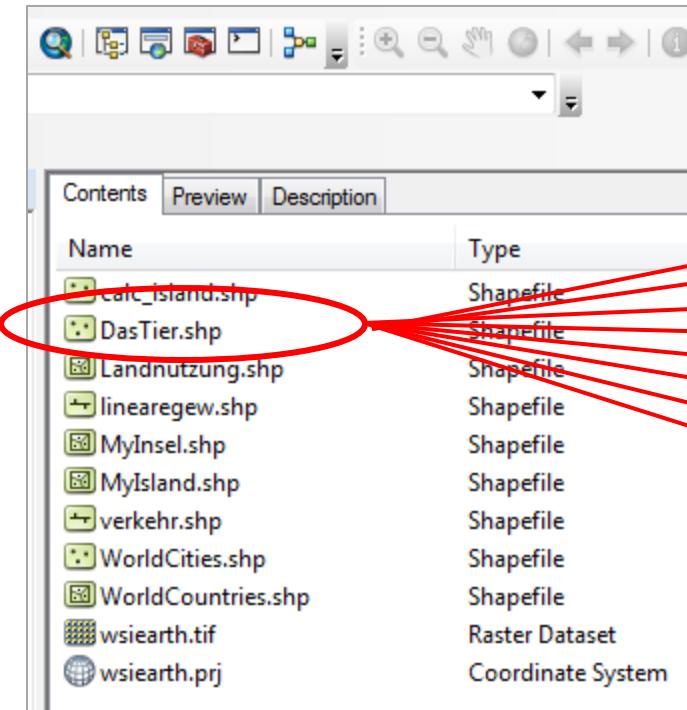
## Per buffer (Clip):

- % land use types in buffer
- road length...

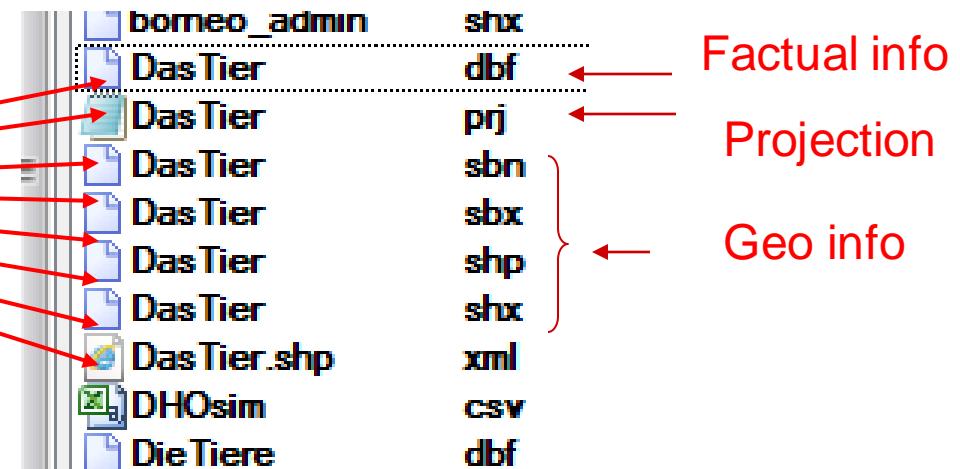
In R:  
gBuffer  
st\_buffer

# ArcCatalog – Data Formats

ArcCatalog



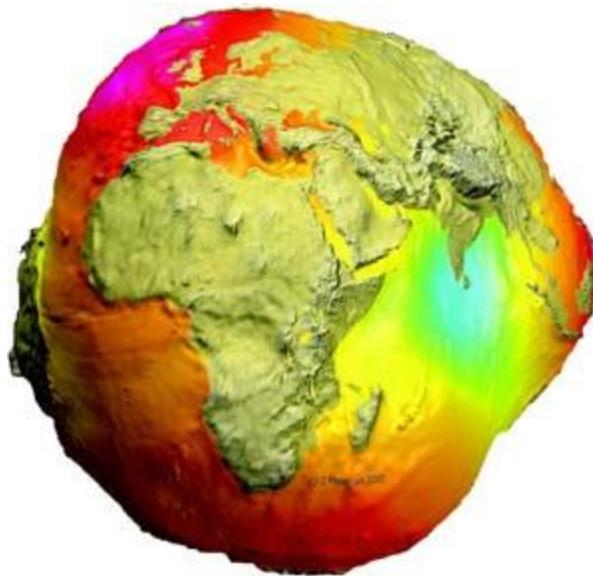
'Normal' Explorer



If you copy-paste-rename spatial polygon layers, you need to do it with all files

- Copying/renaming of data files and moving of projects only via ArcCatalog!

# Coordinate Systems and Projections



„Normaler“ Explorer

m [daten2] 21,070,848 k von 31,457,2	
m:\IZW\GIS_Lab\GIS_Kurs\Liebenthal\**	
Name	Erw.
[..]	
[info]	
[TestKernel]	
Clip_Tmp_Liebenthal	dbf
Clip_Tmp_Liebenthal	prj
Clip_Tmp_Liebenthal	sbn
Clip_Tmp_Liebenthal	sbx
Clip_Tmp_Liebenthal	shp
Clip_Tmp_Liebenthal	shx
Geodatenzentrum	url
GPS_Collar05959_09113..	TXT
GPS_Collar05959_10052..	BTX
kde	aux
Lbt	txt
LiebenthalTest	mxd

How to describe where we are...

```
PROJCS["DHDN_3_Degree_Gauss_Zone_4",GEOGCS["GCS_Deutsche_Hauptdreiecksnetz",DATUM["D_Deutsche_Hauptdreiecksnetz",SPHEROID["Bessel_1841",6377397.155,299.1528128]],PRIMEM[0.0174532925199433],PROJECTION["Gauss_Kruger"],PARAMETER["False_Easting",0.0],PARAMETER["Central_Meridian",12.0],PARAMETER["Scale_Factor",0.9999],UNIT["Meter",1.0]]
```

=WKT well-known textfile  
-> future use in spatial R

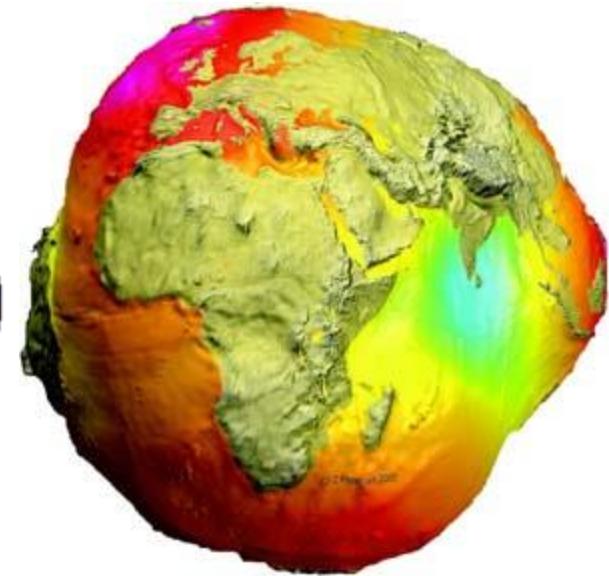
# Where are we? – Reference System



Ball (Sphere)?  
(Pythagoras  
500 BC)



Ellipse?  
That's better!  
(Sir Isaac Newton)

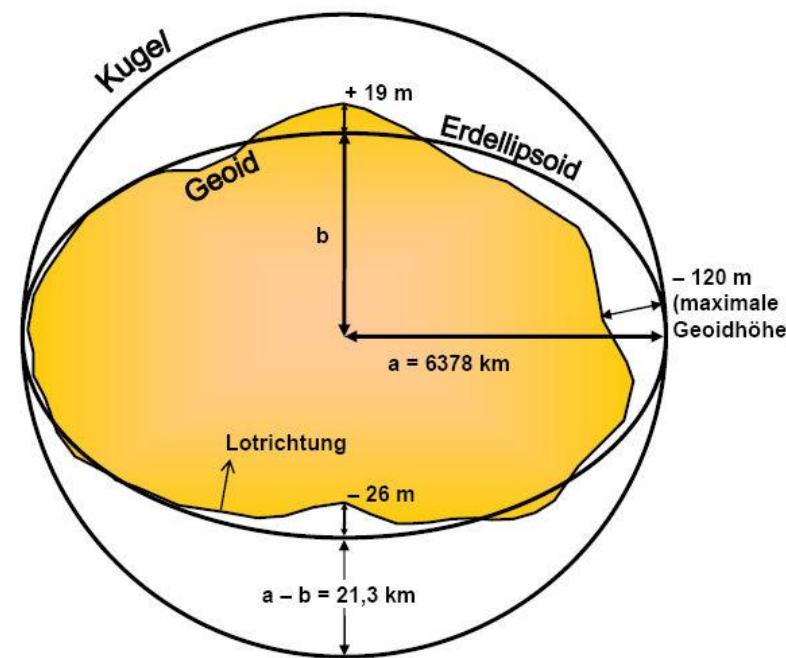
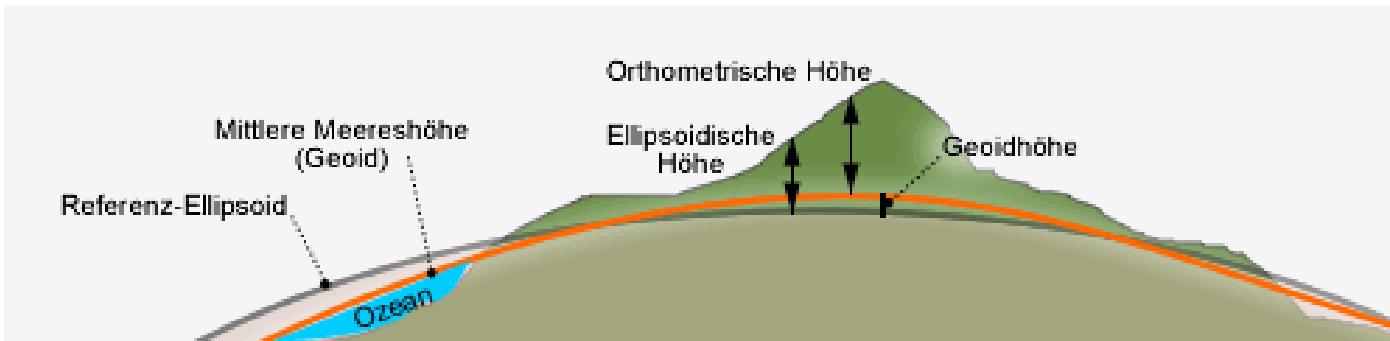


Geoid!  
(Gauß)

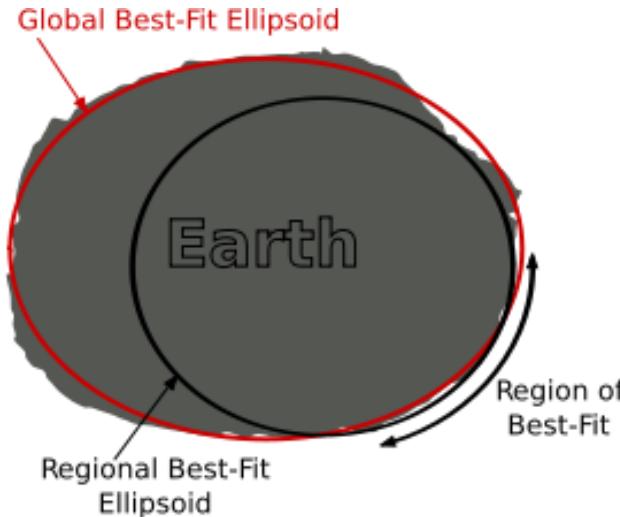
Colour code: deviation  
from reference ellipsoid

# Approximation of a geoid...

Easier to describe ellipsoids mathematically



# Geographic coordinate system (GCS)



Usually given in **angular units**  
(latitude/ longitude)

Also includes:

Ellipsoid = size and shape of ellipse

Ellipsoids (spheroids) mathematically more easily described

Geodetic datum = position and orientation  
of reference ellipsoid

Global Datum: WGS84  
with GRS80-Ellipsoid

GPS !!!

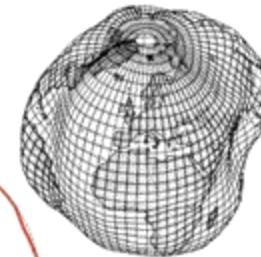
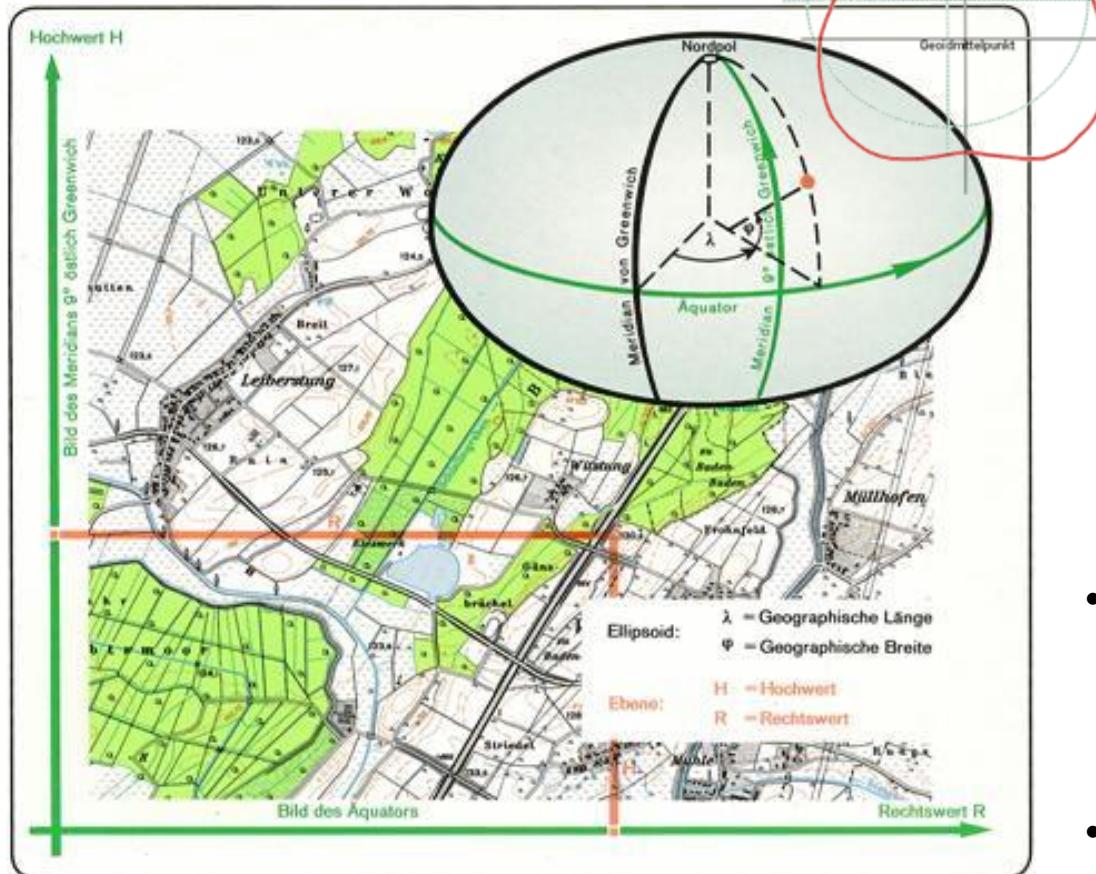
World Geodetic System 1984  
With Geodetic Reference System 1980

Local Datum: eg  
Potsdam (Rauenberg)  
with Bessel-Ellipsoid

# Projected coordinate systems (PCS)

From 3D to 2D

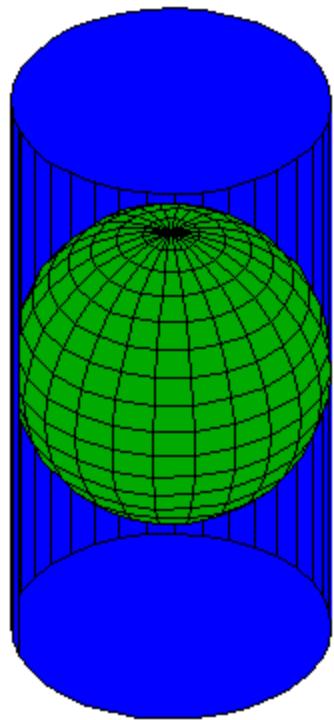
What if I want distances in metres  
or area sizes in m<sup>2</sup> ?



- constant lengths, angles, and areas across the two dimensions
- linear unit of measure

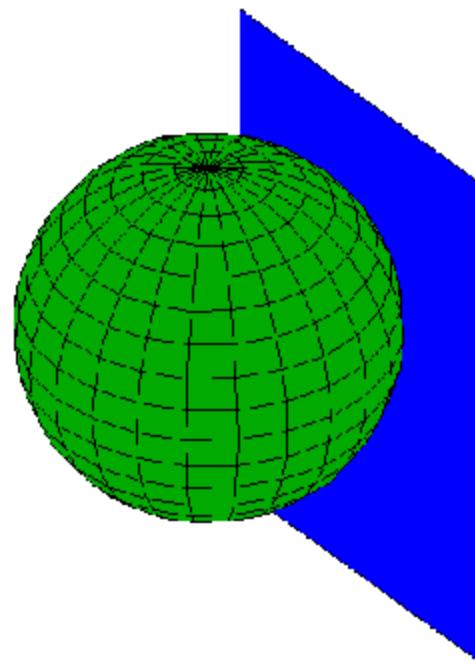
# Projections

cylindric



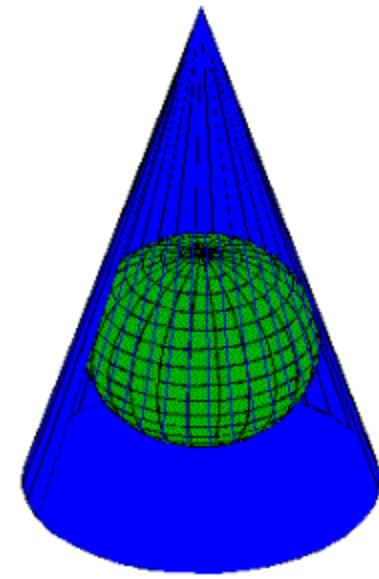
eg UTM

Planar/  
azimuthal



eg Lambert

conic



eg Albers

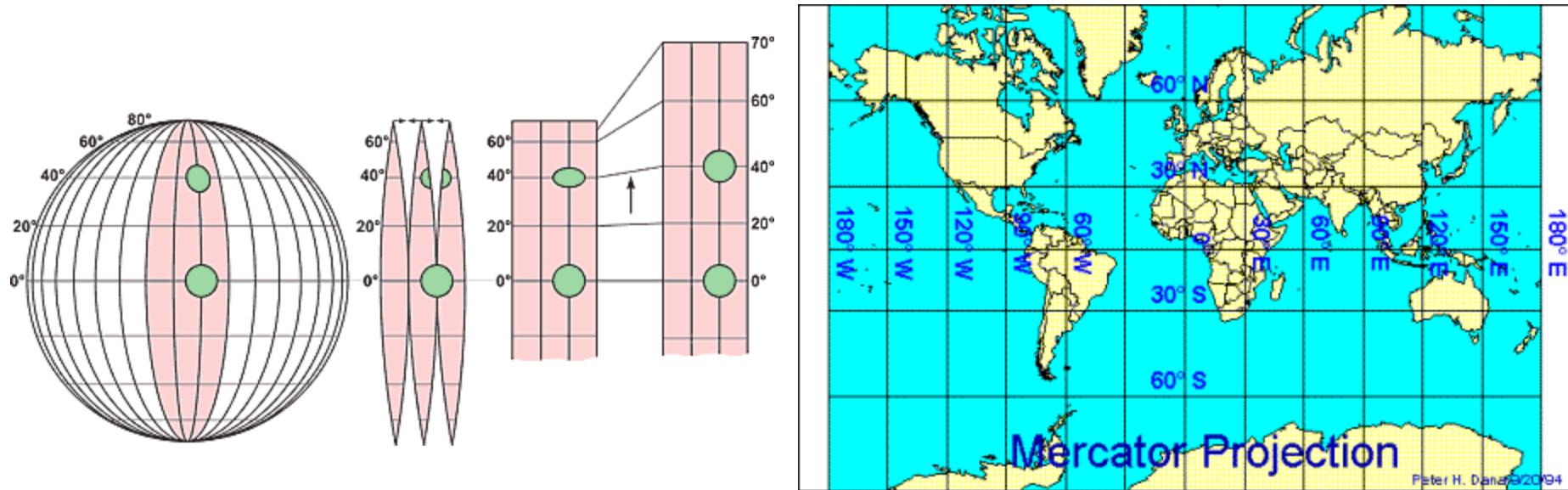
eg Mercator projections (conformal)

Gauss-Krüger (in 3°/ 6° stripes), UTM (equal area-projection)

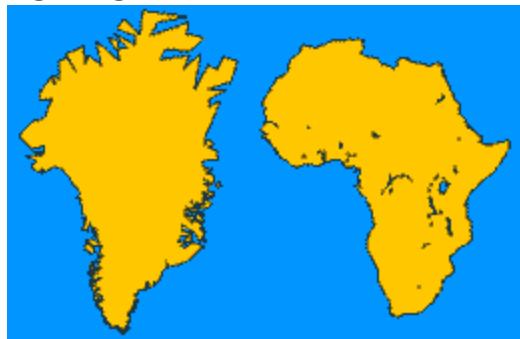
Quelle: Peter H. Dana 1994, Uni Siegen

<http://resources.esri.com/help/9.3/arcgisengine/dotnet/89b720a5-7339-44b0-8b58-0f5bf2843393.htm>

# Example: Cylindrical Projection



geographic



projected



Comparison of size between Greenland and Africa in the Mercator projection which is conformal but not an equal-area projection (left) and in the equal-area Lambert projection (right)

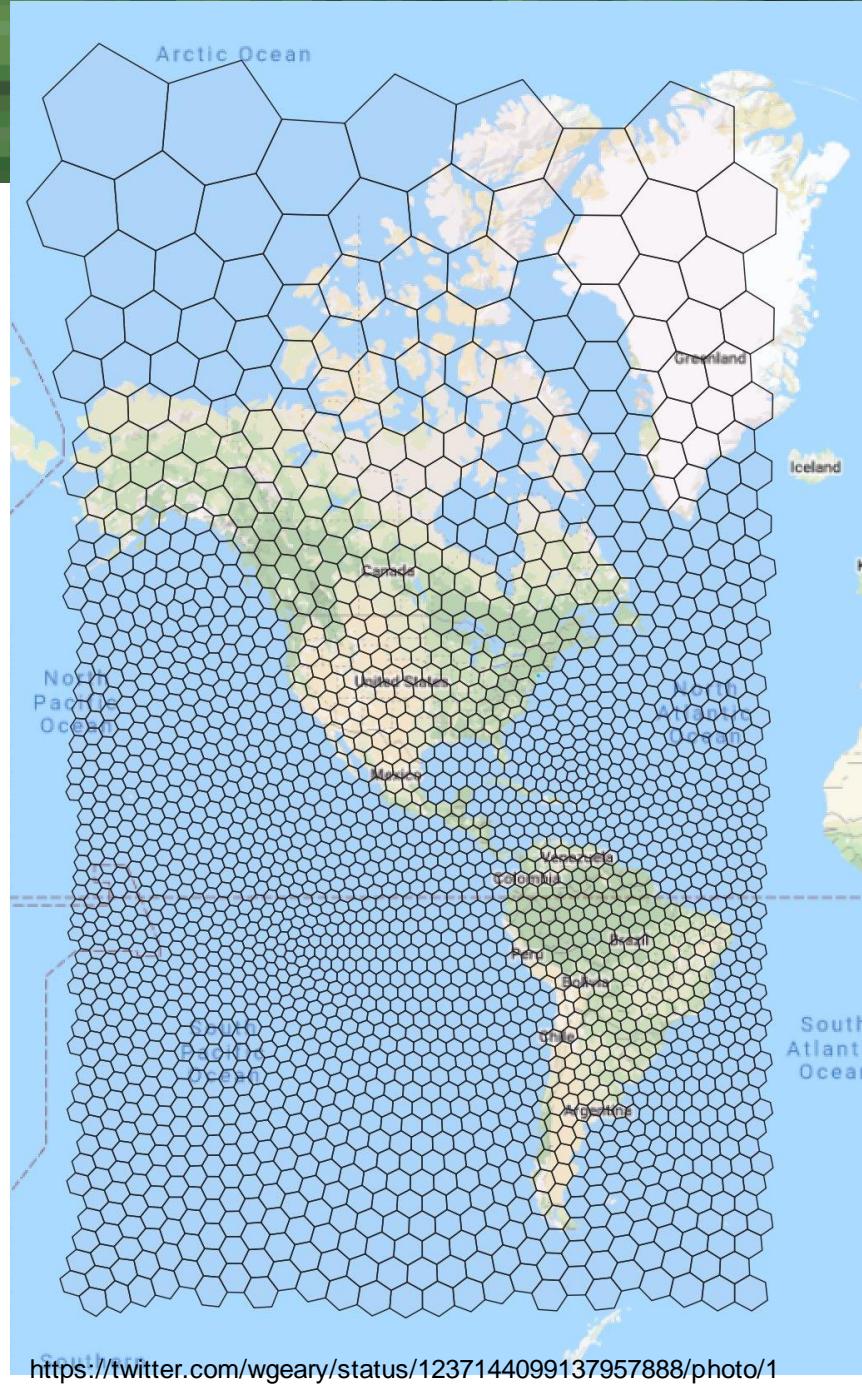
- <https://twitter.com/i/status/1237163708285104129>

World Mercator projection with country going to true size



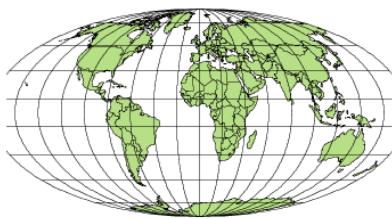
@neilrkaye

<https://www.visualcapitalist.com/map-true-size-of-africa/>

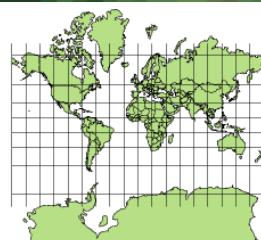


<https://geoawesomeness.com/best-map-projection/>

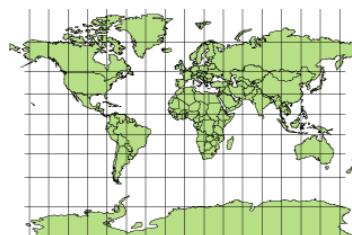
# <http://de.wikipedia.org/wiki/Kartennetzentwurf>



Mollweide-Projektion



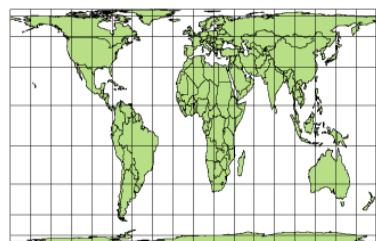
Mercator-Projektion



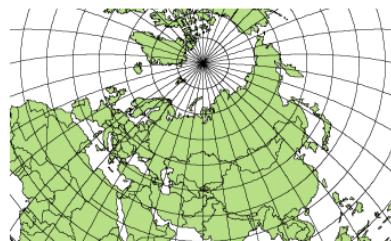
Zylinderprojektion nach Miller



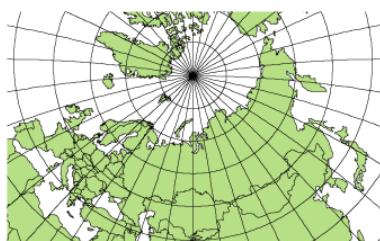
Hammer-Aitoff-Projektion



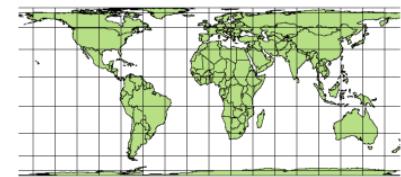
Peters-Projektion



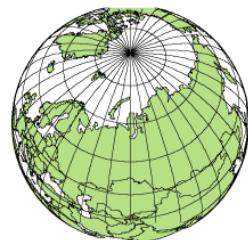
Längentreue Azimutalprojektion



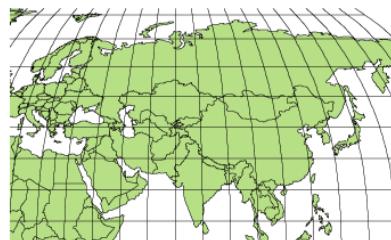
Stereographische Projektion



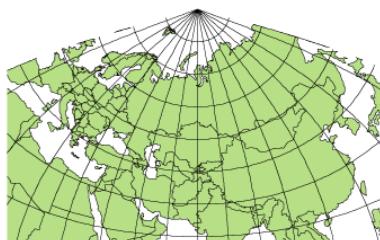
Behrmann-Projektion



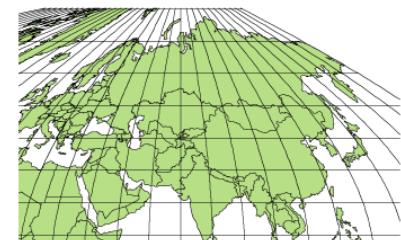
Senkrechte Umgebungsperspektive



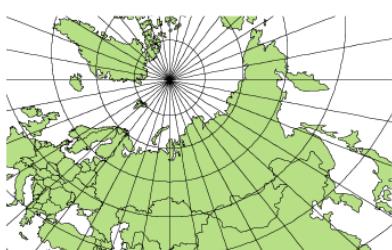
Robinson-Projektion



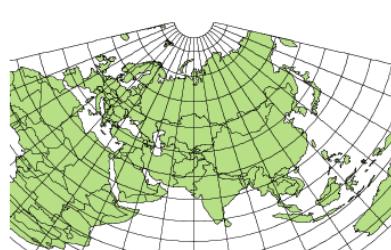
Hotine Oblique Mercator-Projektion



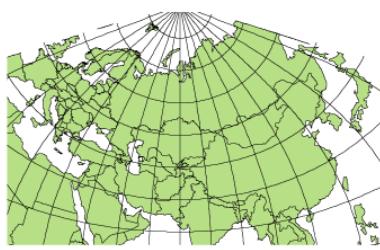
Sinusoidale Projektion



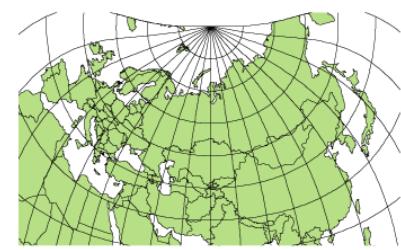
Gnomonische Projektion



Flächentreue Kegelprojektion



Transverse Mercator-Projektion



Cassini-Soldner-Projektion

# Summary

- Ellipsoid: describes form and size of ellipse (Bessel, Clarke, Krassowski...)
- Date/ datum: origin and orientation of the ellipsoid
- Projection: mapping rule concerning depiction of sphere/ellipsoid/geoid in 2D (plane)

Projection-file (.prj) in ArcGIS describes this to be able to display maps of different projections and coordinates on top of each other.

```
PROJCS["DHDN_3_Degree_Gauss_Zone_4",GEOGCS["GCS_Deutsche_Hauptdreiecksnetz",DATUM["D_Deutsche_Hauptdreiecksnetz",SPHEROID["Bessel_1841",6377397.155,299.1528128]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],PROJECTION["Gauss_Kruger"],PARAMETER["False_Easting",4500000.0],PARAMETER["False_Northing",0.0],PARAMETER["Central_Meridian",12.0],PARAMETER["Scale_Factor",1.0],PARAMETER["Latitude_Of_Origin",0.0],UNIT["Meter",1.0]]
```

# In R – crs – coordinate reference system

- EPSG stands for **European Petroleum Survey Group**. They publish a database of coordinate system information plus some very good related documents on map projections and datums.
- <https://support.esri.com/en/technical-article/000002814>
- <http://spatialreference.org/>

In R:

```
{raster}:
raster@crs <- CRS("+init=epsg:4326")
```

```
{terra}
crs(ras_bio_asc_01) <- "+init=epsg:4326"
```

# EPSG code WGS84: 4326

spatialreference.org/ref/epsg/4326/

Spatial Reference

epsg projection 4326 - wgs 84

Home | Upload Your Own | List user-contributed references | List all references

Previous: [EPSG:4324: WGS 72BE](#) | Next: [EPSG:4327: WGS 84 \(geographic 3D\)](#)

[Link to this Page](#)

**EPSG:4326**

WGS 84 ([Google it](#))

- **WGS84 Bounds:** -180.0000, -90.0000, 180.0000, 90.0000
- **Projected Bounds:** -180.0000, -90.0000, 180.0000, 90.0000
- **Scope:** Horizontal component of 3D system. Used by the GPS satellite navigation system and for NATO military geodetic surveying.
- **Last Revised:** Aug. 27, 2007
- **Area:** World

+proj=longlat +ellps=WGS84 +datum=WGS84 +no\_defs

(soon migration to WKT file in spatial R)

In R:

```
{raster}:
raster@crs <- CRS("+init=epsg:4326")
```

```
{terra}
```

```
crs(ras_bio_asc_01)<- "+init=epsg:4326"
```

# Common CRS errors – assignment & projection/ transformation

Data.frame 1	Longitude 13.0	Latitude 52.0
Data.frame 2	X 390244.32	Y 5823218.29

**ASSIGN  
the CRS:  
`st_crs()`**



# Common CRS errors – assignment & projection/ transformation

Data.frame 1      Longitude

13.0

Latitude

52.0

Data.frame 2      X

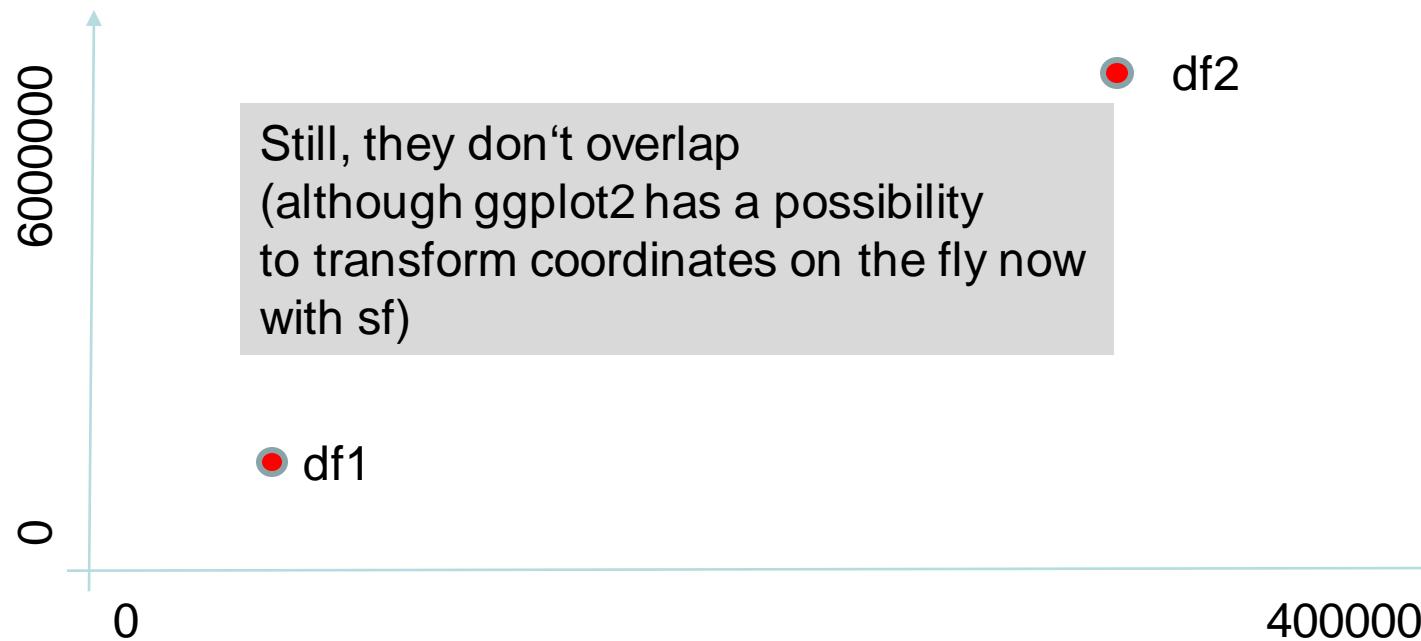
390244.32

Y

5823218.29

WGS84 – angular ( ° )

LAEA 32633 – planar (m)



# Common CRS errors – assignment & projection/ transformation

Data.frame 1      Longitude

13.0

Latitude

52.0

Data.frame 2      X

390244.32

Y

5823218.29

WGS84 – angular ( ° )

LAEA 32633 – planar (m)

6000000

0

0

df1

df2

Still, they don't overlap  
(although ggplot2 has a possibility  
to transform  
with sf)

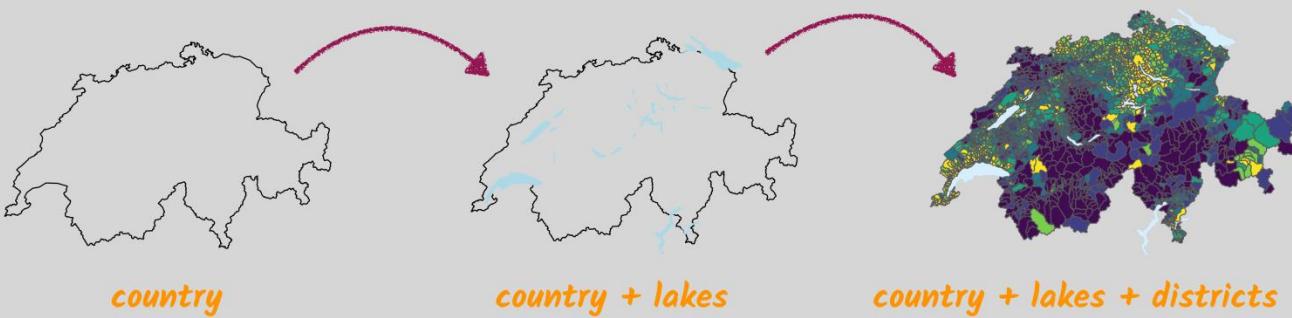
**TRANSFORM/ PROJECT  
one CRS into the other:  
`st_transform()/ project()`  
Then the coords change!**

# Visualisation in R



- <https://drive.google.com/file/d/1de3dTrAj90e2R9zp9gYPy2kdCjbzdvvK/view>
- Slide by Giulia Ruggeri

## {ggplot2} : geom\_sf()

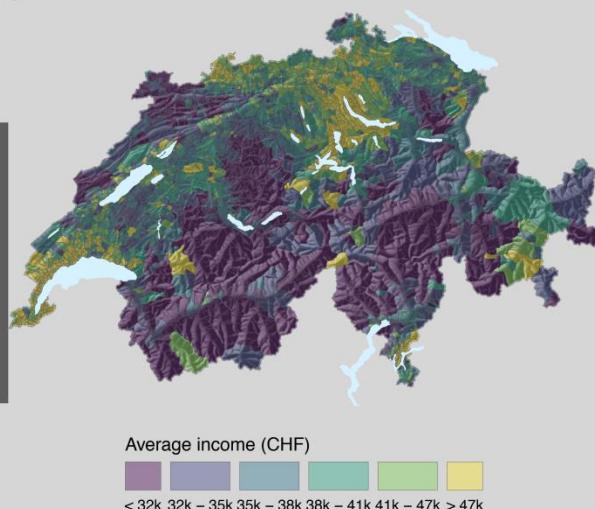


```
# plot the data  
> ggplot() +  
  geom_sf(data = country) +  
  geom_sf(data = lakes, fill = "#D6F1FF") +  
  geom_sf(data = districts,  
         aes(fill = income_categories))
```

## {ggplot2} : geom\_raster() + geom\_sf()

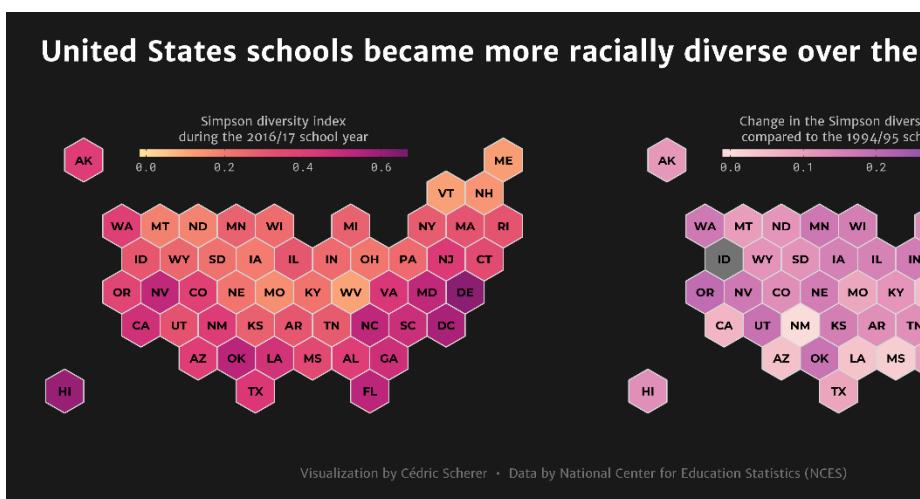
```
> ggplot() +  
  geom_raster(data = relief,  
              aes(x = x,  
                  y = y,  
                  alpha = relief)) +  
  geom_sf(data = country) +  
  geom_sf(data = lakes, fill = "#D6F1FF") +  
  geom_sf(data = districts,  
         aes(fill = income_categories))
```

To be done in the hands-on R course.

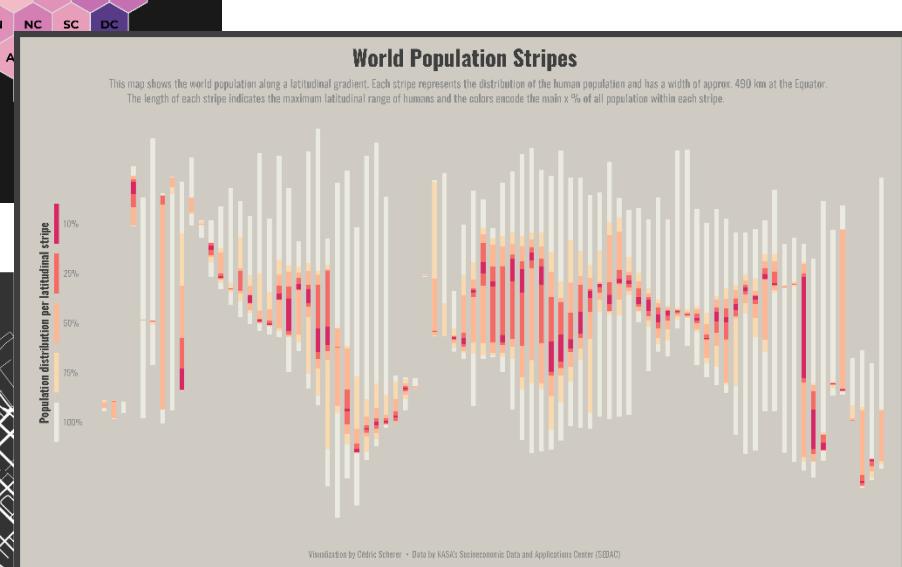
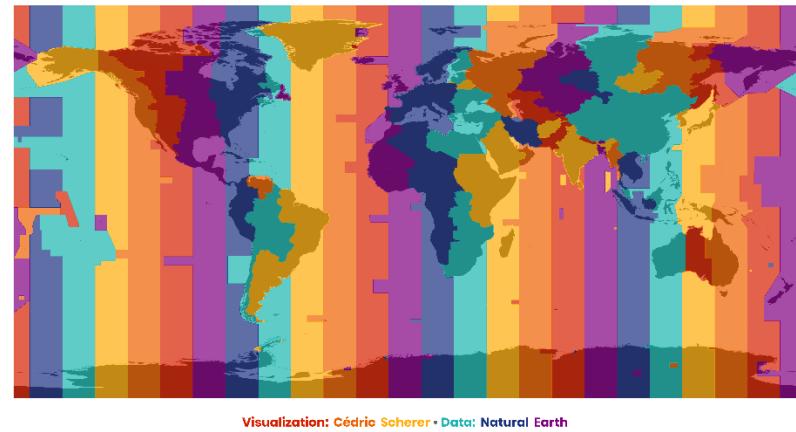


# Viz: Get inspired by Cédric & ggplot2

<https://www.data-vizard.com/>



## The Time Zones of the World



# Summary: The main commands

## Vector data

**sf**

st\_read / st\_as\_sf  
st\_crs()  
st\_transform()

**sp**

readOGR()  
proj4string()  
spTransform()

as(x, 'Spatial')

as(x, 'sf')

st\_write()

writeOGR()

## Raster data

**raster**

raster()  
crs()  
projectRaster()

**terra**

rast()  
crs()  
project()

writeRaster()

*Data manipulation similar to base R /  
(data.frame, matrix, list, S4)*

# Summary: The main commands

## Vector data

**sf**

st\_read / st\_as\_sf  
st\_crs()  
st\_transform()

**sp**

readOGR()  
proj4string()  
spTransform()

## Raster data

**raster**

raster()  
crs()  
projectRaster()

**TAKE HOME MESSAGE:** spatial R is the same as handling basic data types (data.frame, matrix, list)  
– just that you need to understand the added spatial component handling (*geometry*)

Use the course scripts for quick lookup (like a vocabulary book....) and make your own notes



Practical things...

# CHEATSHEETS

# Base R Cheat Sheet

## Getting Help

### Accessing the help files

?mean

Get help of a particular function.

help.search('weighted mean')

Search the help files for a word or phrase.

help(package = 'dplyr')

Find help for a package.

### More about an object

str(iris)

Get a summary of an object's structure.

class(iris)

Find the class an object belongs to.

## Using Libraries

install.packages('dplyr')

Download and install a package from CRAN.

library(dplyr)

Load the package into the session, making all its functions available to use.

dplyr::select

Use a particular function from a package.

data(iris)

Load a built-in dataset into the environment.

## Working Directory

getwd()

Find the current working directory (where inputs are found and outputs are sent).

setwd('C://file/path')

Change the current working directory.

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

Vectors			Programming																
Creating Vectors			For Loop			While Loop													
c(2, 4, 6)	2 4 6	Join elements into a vector	for (variable in sequence){	Do something	}	while (condition){	Do something	}											
2:6	2 3 4 5 6	An integer sequence	Example																
seq(2, 3, by=0.5)	2.0 2.5 3.0	A complex sequence	for (i in 1:4){ j <- i + 10 print(j) }																
rep(1:2, times=3)	1 2 1 2 1 2	Repeat a vector																	
rep(1:2, each=3)	1 1 1 2 2 2	Repeat elements of a vector																	
Vector Functions																			
sort(x)	rev(x)	Return x sorted. Return x reversed.	if (condition){ Do something } else { Do something different }	If Statements			Functions												
table(x)	unique(x)	See counts of values. See unique values.	Example																
Selecting Vector Elements																			
By Position																			
x[4]	The fourth element.							Example											
x[-4]	All but the fourth.							square <- function(x){ squared <- x*x return(squared) }											
x[2:4]	Elements two to four.																		
x[-(2:4)]	All elements except two to four.																		
x[c(1, 5)]	Elements one and five.																		
By Value																			
x[x == 10]	Elements which are equal to 10.							Input											
x[x < 0]	All elements less than zero.							Output											
x[x %in% c(1, 2, 5)]	Elements in the set 1, 2, 5.							Description											
Named Vectors																			
x['apple']	Element with name 'apple'.							df <- read.table('file.txt') write.table(df, 'file.txt')											
Conditions			a == b	Are equal	a > b	Greater than	a >= b	Greater than or equal to											
			a != b	Not equal	a < b	Less than	a <= b	Less than or equal to											
								is.na(a)											
								is.null(a)											
								is.na(b)											
								is.null(b)											
Reading and Writing Data																			
Input			Output			Description													
df <- read.table('file.txt')			write.table(df, 'file.txt')			Read and write a delimited text file.													
df <- read.csv('file.csv')			write.csv(df, 'file.csv')			Read and write a comma separated value file. This is a special case of read.table/write.table.													
load('file.RData')			save(df, file = 'file.Rdata')			Read and write an R data file, a file type special for R.													

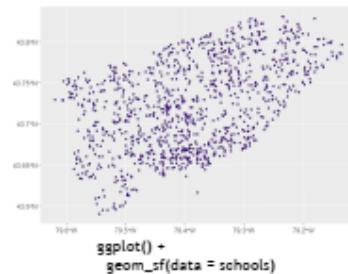
# Spatial manipulation with sf: : CHEAT SHEET

The sf package provides a set of tools for working with geospatial vectors, i.e. points, lines, polygons, etc.



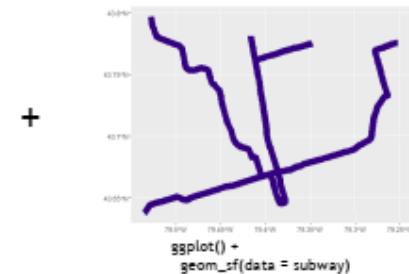
## Geometric confirmation

- ▢ ⊕ st\_contains(x, y, ...) Identifies if x is within y (i.e. point within polygon)
- ▢ ⊕ st\_covered\_by(x, y, ...) Identifies if x is completely within y (i.e. polygon completely within polygon)
- ▢ ⊕ st\_covers(x, y, ...) Identifies if any point from x is outside of y (i.e. polygon outside polygon)
- ▢ ⊕ st\_crosses(x, y, ...) Identifies if any geometry of x have commonalities with y
- ▢ ⊕ st\_disjoint(x, y, ...) Identifies when geometries from x do not share space with y
- ▢ ⊕ st\_equals(x, y, ...) Identifies if x and y share the same geometry
- ▢ ⊕ st\_intersects(x, y, ...) Identifies if x and y geometry share any space
- ▢ ⊕ st\_overlaps(x, y, ...) Identifies if geometries of x and y share space, are of the same dimension, but are not completely contained by each other
- ▢ ⊕ st\_touches(x, y, ...) Identifies if geometries of x and y share a common point but their interiors do not intersect
- ▢ ⊕ st\_within(x, y, ...) Identifies if x is in a specified distance to y



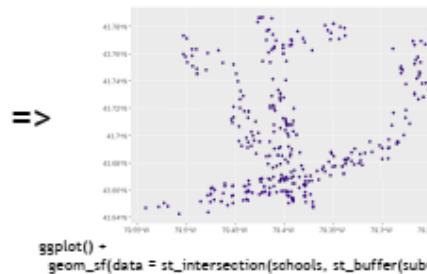
## Geometric operations

- ▢ ⊕ st\_boundary(x) Creates a polygon that encompasses the full extent of the geometry
- ▢ ⊕ st\_buffer(x, dist, nQuadSegs) Creates a polygon covering all points of the geometry within a given distance
- ▢ ⊕ st\_centroid(x, ..., of\_largest\_polygon) Creates a point at the geometric centre of the geometry
- ▢ ⊕ st\_convex\_hull(x) Creates geometry that represents the minimum convex geometry of x
- ▢ ⊕ st\_line\_merge(x) Creates linestring geometry from sewing multi linestring geometry together
- ▢ ⊕ st\_node(x) Creates nodes on overlapping geometry where nodes do not exist
- ▢ ⊕ st\_point\_on\_surface(x) Creates a point that is guaranteed to fall on the surface of the geometry
- ▢ ⊕ st\_polygonize(x) Creates polygon geometry from linestring geometry
- ▢ ⊕ st\_segmentize(x, dfMaxLength, ...) Creates linestring geometry from x based on a specified length
- ▢ ⊕ st\_simplify(x, preserveTopology, dTolerance) Creates a simplified version of the geometry based on a specified tolerance



## Geometry creation

- ▢ ⊕ st\_triangulate(x, dTolerance, bOnlyEdges) Creates polygon geometry as triangles from point geometry
- ▢ ⊕ st\_voronoi(x, envelope, dTolerance, bOnlyEdges) Creates polygon geometry covering the envelope of x, with x at the centre of the geometry
- ▢ ⊕ st\_point(x, c(numeric vector), dim = "XYZ") Creating point geometry from numeric values
- ▢ ⊕ st\_multipoint(x = matrix(numeric values in rows), dim = "XYZ") Creating multi point geometry from numeric values
- ▢ ⊕ st\_linestring(x = matrix(numeric values in rows), dim = "XYZ") Creating linestring geometry from numeric values
- ▢ ⊕ st\_multilinestring(x = list(numeric matrices in rows), dim = "XYZ") Creating multi linestring geometry from numeric values
- ▢ ⊕ st\_polygon(x = list(numeric matrices in rows), dim = "XYZ") Creating polygon geometry from numeric values
- ▢ ⊕ st\_multipolygon(x = list(numeric matrices in rows), dim = "XYZ") Creating multi polygon geometry from numeric values



# Spatial manipulation with sf: : CHEAT SHEET

The sf package provides a set of tools for working with geospatial vectors, i.e. points, lines, polygons, etc.



## Geometry operations

- ⇒ `st_contains(x, y, ...)` Identifies if x is within y (i.e. point within polygon)
- ⇒ `st_crop(x, y, ..., xmin, ymin, xmax, ymax)` Creates geometry of x that intersects a specified rectangle
- ⇒ `st_difference(x, y)` Creates geometry from x that does not intersect with y
- ⇒ `st_intersection(x, y)` Creates geometry of the shared portion of x and y
- ⇒ `st_sym_difference(x, y)` Creates geometry representing portions of x and y that do not intersect
- ⇒ `st_snap(x, y, tolerance)` Snap nodes from geometry x to geometry y
- ⇒ `st_union(x, y, ..., by_feature)` Creates multiple geometries into a single geometry, consisting of all geometry elements

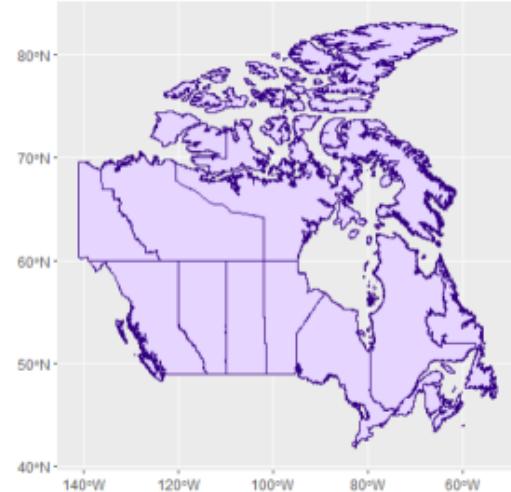
## Misc operations

- ⇒ `st_as_sf(x, ...)` Create a sf object from a non-geospatial tabular data frame
- ⇒ `st_cast(x, to, ...)` Change x geometry to a different geometry type
- ⇒ `st_coordinates(x, ...)` Creates a matrix of coordinate values from x
- ⇒ `st_crs(x, ...)` Identifies the coordinate reference system of x
- ⇒ `st_join(x, y, join, FUN, suffix, ...)` Performs a spatial left or inner join between x and y
- ⇒ `st_make_grid(x, cellsize, offset, n, crs, what)` Creates rectangular grid geometry over the bounding box of x
- ⇒ `st_nearest_feature(x, y)` Creates an index of the closest feature between x and y
- ⇒ `st_nearest_points(x, y, ...)` Returns the closest point between x and y
- ⇒ `st_read(dsn, layer, ...)` Read file or database vector dataset as a sf object
- ⇒ `st_transform(x, crs, ...)` Convert coordinates of x to a different coordinate reference system

## Geometric measurement

- ⇒ `st_area(x)` Calculate the surface area of a polygon geometry based on the current coordinate reference system
- ⇒ `st_distance(x, y, ..., dist_fun, by_element, which)` Calculates the 2D distance between x and y based on the current coordinate system
- ⇒ `st_length(x)` Calculates the 2D length of a geometry based on the current coordinate system

Coordinates projection on the fly



# Free geodata

- <https://earthdata.nasa.gov/learn/toolkits/biological-diversity>

The screenshot shows the homepage of the Earthdata website. At the top, there is a navigation bar with icons for user profile, search, and login. Below the header, the main title 'BIOLOGICAL DIVERSITY' is prominently displayed over a background image of a dense tropical forest. A search bar with the placeholder 'Search datasets, news, articles, and information' and a magnifying glass icon is located below the title. The page includes a breadcrumb menu ('Learn > Data Toolkits > Biological Diversity') and social media sharing buttons (Twitter, Facebook, Pinterest).

Biological diversity, or biodiversity, refers to the variety of all life on Earth—from genes to species, ecosystems, and biomes. Research has shown that global biodiversity has been on the decline. NASA studies how and why global biodiversity is changing, and the effects of these changes on and interactions with Earth's interrelated systems. Sensors on a suite of NASA satellites, combined with airborne platforms, in situ observations and models, provide measurements of biodiversity and environmental variables such as vegetation productivity, biomass, habitat suitability, land cover and land use change,

aquatic ecology and human interactions with the environment. These long-term observations and models enable scientists to better understand Earth's global biodiversity and how it is changing.

This toolkit is designed to support research into biodiversity by providing easy access to data and other resources on these topics:

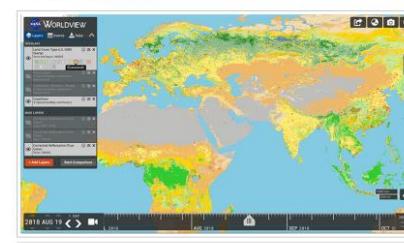
[Vegetation](#) | [Species Distribution](#) | [Human Dimensions](#) | [Habitat Suitability](#) | [Aquatic Ecology](#)

## Discover Data



Aqua Moderate Resolution Imaging Spectroradiometer (MODIS) Vegetation indices data from Earthdata Search. Earthdata Search is a data discovery and data access application that enables access to the NASA Earth Observing System Data and Applications System (EOSDIS) Earth science data across the Distributed Active Archive Centers (DAACs).

## Visualize Data



Visualization of the Terra and Aqua Moderate Resolution Imaging Spectroradiometer (MODIS) Land Cover Type in Worldview. The EOSDIS Worldview mapping application provides the capability to interactively browse global, full-resolution satellite imagery layers.

## Discover Biodiversity Data

### Vegetation

Vegetation is a primary component of terrestrial biodiversity, playing a critical role in the energy budget and in many of our biogeochemical cycles. Maintaining species richness ensures the productivity



### Habitat Suitability

Habitat suitability serves as a proxy for species distribution. Models are used to estimate the potentiality of a habitat for a given species by integrating environmental variables, such as land cover type, temperature, precipitation, soil moisture,



# Free data in R

- rnaturalearth
- terrainr (and elevatr, rayvista,...)
- osmdata

Tutorial Part II – R goes spatial with terra

Introduction  
Basics  
Raster data  
Vector data / shapefiles

Data Sources

- {rnaturalearth}
- {osmdata}
- {elevatr}

Advanced Map with {tmap}

Exercise 1  
END

We can quickly plot it:

```
ggplot(world) + geom_sf(aes(fill = economy)) + theme_void()
```

1. Developed region: G7  
2. Developed region: nonG7  
3. Emerging region: BRIC  
4. Emerging region: MIFT  
5. Emerging region: G20  
6. Developing region  
7. Least developed region

You can specify the scale, category and type you want as in the examples below.

```
glacier_small <- ne_download(type = "glaciered_areas", category = "physical",  
                             scale = "small", returnclass = "sf")
```

```
## OGR data source with driver: ESRI Shapefile  
## Source: "C:\Users\kramer\AppData\Local\Temp\RtmpcnF2gA", layer: "ne_110m_glaciered_areas"  
## with 11 features  
## It has 3 fields  
## Integer64 fields read as strings:  scalerank
```

```
glacier_large <- ne_download(type = "glaciered_areas", category = "physical",  
                             scale = "large", returnclass = "sf")
```

Stephanie Kramer-Schadt  
Last Update March 7, 2022

Cedric has added them to the course 2 file!



Leibniz Institute for Zoo  
and Wildlife Research  
IN THE FORSCHUNGSVERBUND BERLIN E.V.

Member of the  
  
Leibniz Association

# Let's start



Leibniz Institute for Zoo  
and Wildlife Research  
IN THE FORSCHUNGSVERBUND BERLIN E.V.

Member of the  
  
Leibniz Association

# Exercises 1 (pre-course preparation)

# RMarkdown

Rstudio -> File -> New File -> R Markdown... -> ok

Tutorial Part II – R goes spatial with terra

Introduction  
Basics  
Raster data  
Vector data / shapefiles  
**Data Sources**  
  {rnaturalearth}  
  {osmdata}  
  {elevatr}

Advanced Map with `(tmpr)`

Exercise 1  
END

We can quickly plot it:

```
## [56] "region_un" "subregion" "region_kb" "name_len" "long_len"  
## [61] "abbrev_len" "tiny" "homepart" "geometry"
```

ggplot(world) + geom\_sf(aes(fill = economy)) + theme\_void()

economy

1. Developed region: G7
2. Developed region: nonG7
3. Emerging region: BRIC
4. Emerging region: MIKT
5. Emerging region: G20
6. Developing region
7. Least developed region

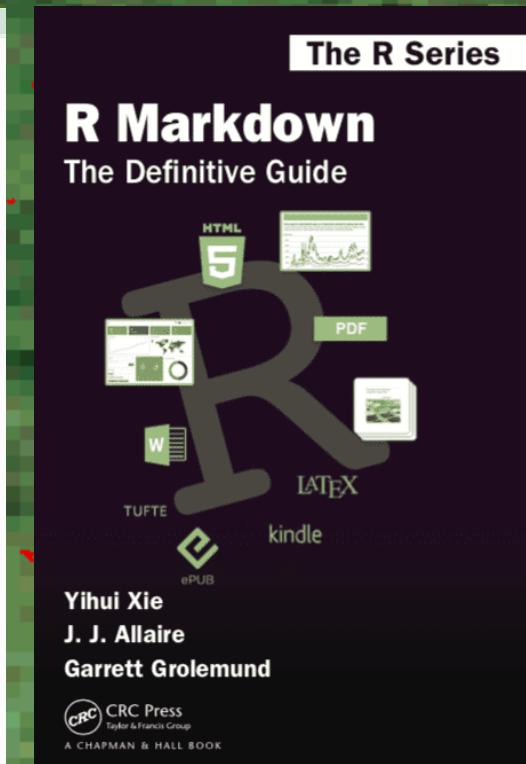
You can specify the scale, category and type you want as in the examples below.

```
glacier_small <- ne_download(type = "glaciated_areas", category = "physical",  
                           scale = "small", returnclass = "sf")
```

```
## OGR data source with driver: ESRI Shapefile  
## Source: "C:\Users\kramer\AppData\Local\Temp\Rtmpcnf2gA", layer: "ne_110m_glaciated_areas"  
## with 11 features  
## It has 3 fields  
## Integer64 fields read as strings: scalarank
```

```
glacier_large <- ne_download(type = "glaciated_areas", category = "physical",  
                           scale = "large", returnclass = "sf")
```

Stephanie Kramer-Schadt  
Last Update March 7, 2022

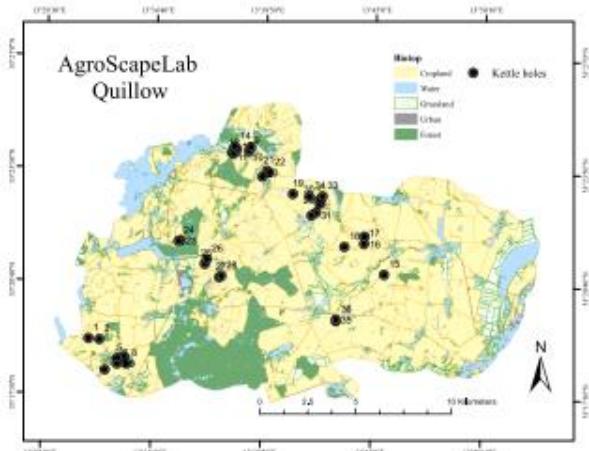


<https://bookdown.org/yihui/rmarkdown/>

## Habitat quality and connectivity in kettle holes enhance bee diversity in agricultural landscapes

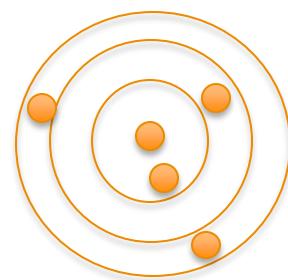
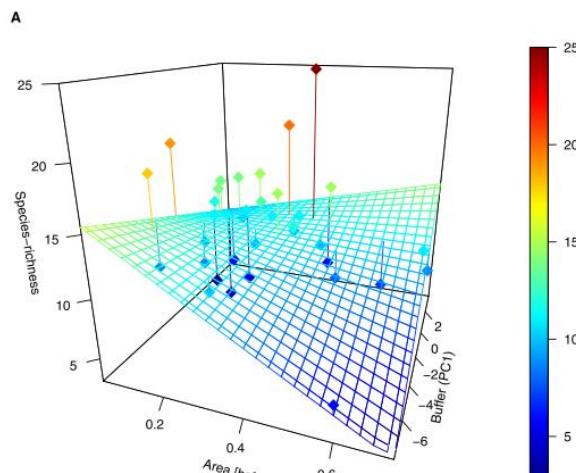


Sissi Lozada-Gobilard <sup>a,b,\*<sup>1,2</sup></sup>, Carlos Miguel Landivar Albis <sup>c</sup>, Karolina Beata Rupik <sup>d</sup>, Marlene Pätzig <sup>e,<sup>3</sup></sup>, Sebastian Hausmann <sup>f,g</sup>, Ralph Tiedemann <sup>b</sup>, Jasmin Joshi <sup>g,h,<sup>4</sup></sup>



- Kettle hole density in different buffers
- Number of habitat types around kettle holes
- Tree cover density

*S. Lozada-Gobilard et al.*



# Exercises

- [https://github.com/stephkramer/Course2\\_RSpatial](https://github.com/stephkramer/Course2_RSpatial)

README.md



Using our long standing projects in Borneo, we demonstrate how to use R as a GIS. A precondition for following this course is knowing well the basics of R, as summarised in [R Intro](#). Please download the course data here (approx. 600 MB):

<https://www.dropbox.com/sh/91qey7u5ohn5fmf/AAD84ZxnExw4Chd93XidEaF6a?dl=0>

and then follow the course here: [R goes Spatial](#)

As a 'learning control exercise', please solve Exercise 1 in course 2 material yourself (C2\_E1).

More exercises here:

- Course 2 Exercise 2 [C2\\_E2](#)
- Course 2 Exercise 3 [C2\\_E3](#)

# Let's start in Breakout Rooms

## Course 2 Exercise 2

### Exercise 2.1

Using data on green spaces in Berlin, estimate the number of wild boars observed inside green areas that are 1 ha or larger. What is the proportion of wild boars outside large green areas? Visualize the observations on a map showing the green spaces and the observations encoded by their location inside or outside the green spaces.

Individual Steps:

- 1. Store the simple feature data `sf_green` which is provided by the `{d6berlin}` package in a local object.
- 2. Calculate the area for each green space polygon and filter for green spaces that are equal or greater than 1 ha.
- 3. Load the wild boar data (`data_wb_melden_en.csv`) and turn it into a simple feature object. Remember to set the correct CRS!
- 4. Intersect the wild boar data with the filtered green spaces and extract the number of unique observations.
- 5. Use the new data to visualize observations inside and outside the green spaces that are 1 ha or larger as a map.



Leibniz Institute for Zoo  
and Wildlife Research  
IN THE FORSCHUNGSVERBUND BERLIN E.V.

Member of the  
  
Leibniz Association

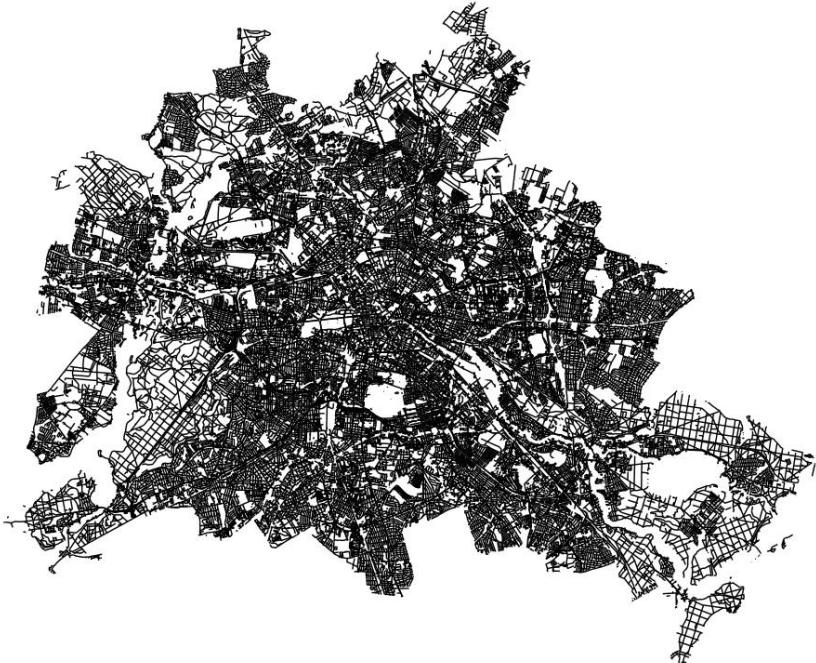
# Course 2 Exercise 2

- Little helpers for projects in Berlin:
- d6berlin

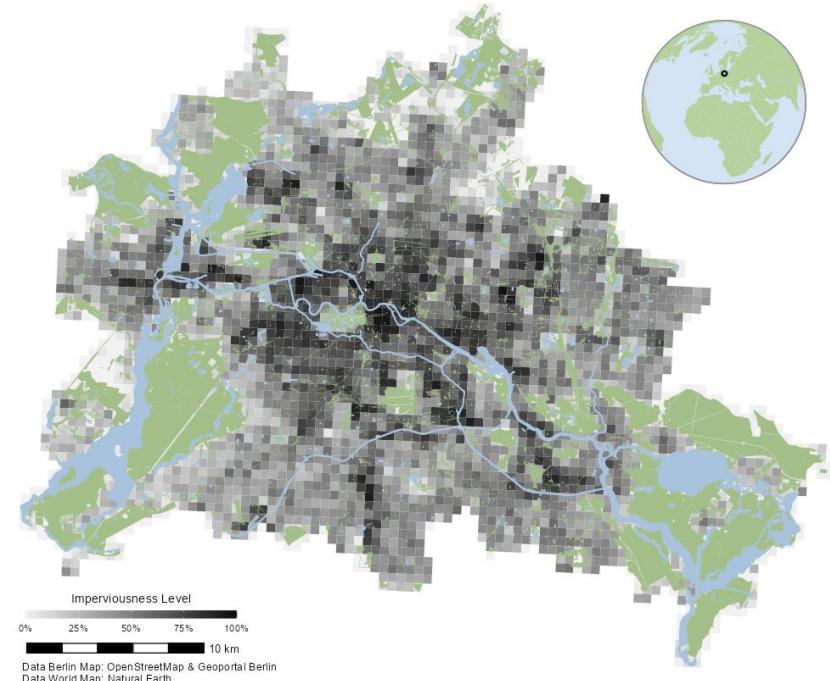
# The {d6berlin} R Package

The {d6berlin} package aims to provide spatial data and template maps for Berlin

`d6berlin::sf_roads`



`d6berlin::base_map_imp()`



# The {d6berlin} R Package

The {d6berlin} package aims to provide  
**spatial data and template maps for Berlin.**

```
install.packages("devtools")
devtools::install_github("EcoDynIZW/d6berlin")
```



```
library(d6berlin)
names(sf_green)
[1] "osm_id" "code" "fclass" "name" "district_name" "district_id" "district_key" "geometry"

unique(sf_green$fclass)
[1] "scrub" "grass" "forest" "meadow" "nature_reserve" "beach" "heath" "cliff"

sf_forest <- subset(sf_green, fclass == "forest")
```

# The {d6berlin} R Package

The {d6berlin} package aims to provide  
**spatial data and template maps for Berlin.**

```
install.packages("devtools")
devtools::install_github("EcoDynIZW/d6berlin")
```

```
library(d6berlin)
library(ggplot2)

ggplot(sf_forest) + geom_sf()
```



# The {d6berlin} R Package

The {d6berlin} package aims to provide  
**spatial data and template maps for Berlin.**

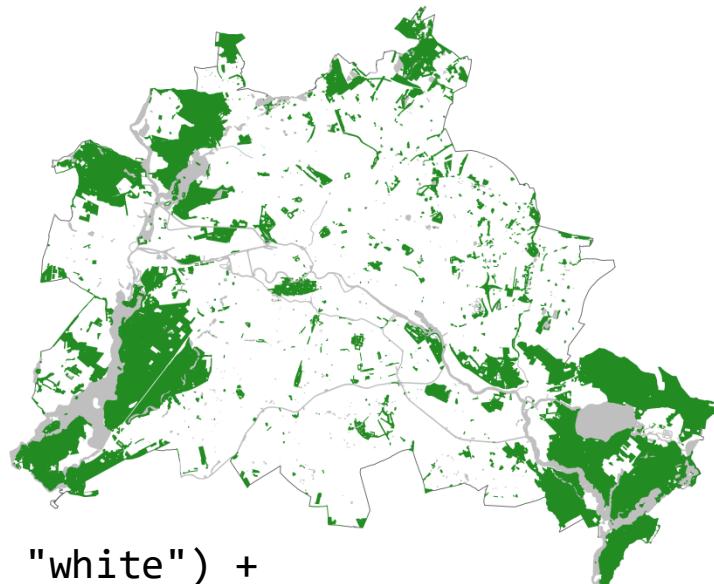
```
install.packages("devtools")
devtools::install_github("EcoDynIZW/d6berlin")
```



```
library(d6berlin)
library(ggplot2)

ggplot(sf_forest) + geom_sf()

ggplot() +
  geom_sf(data = sf_berlin, fill = "white") +
  geom_sf(data = sf_water, fill = "grey75", color = "grey75") +
  geom_sf(data = sf_forest, fill = "forestgreen", color = "forestgreen")
```



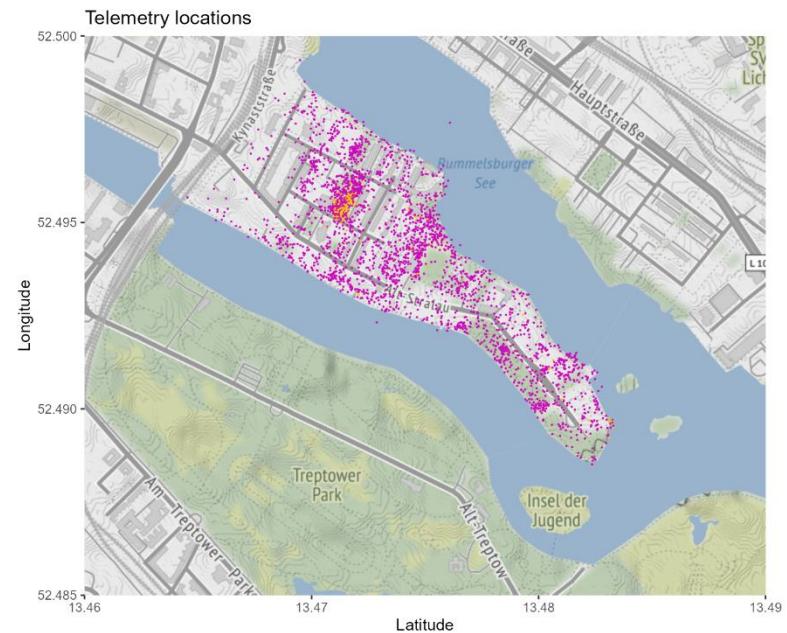


Leibniz Institute for Zoo  
and Wildlife Research  
IN THE FORSCHUNGSVERBUND BERLIN E.V.

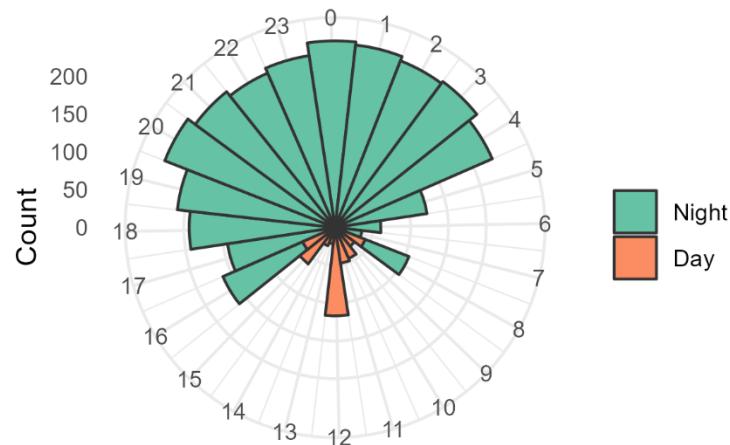
Member of the



# Sneek preview: Day 2 Introduction to movement analysis



Events by Time of the Day



# Day 2: Intro to Movement Data

[https://github.com/stephkramer/Course4\\_MoveQ](https://github.com/stephkramer/Course4_MoveQ)

Clone or download and unzip the repository

The screenshot shows a GitHub repository page for 'Course4\_MoveQ'. The repository is public and has 1 branch and 0 tags. The code tab is selected. A context menu is open over the 'Clone' button, displaying options for 'Clone' (HTTPS, SSH, GitHub CLI), 'Open with GitHub Desktop', and 'Download ZIP'. The repository description is 'Introduction to movement data analysis'. It has 0 stars, 2 watching, and 0 forks. There are no packages published.

Code

main · 1 branch · 0 tags

stephkramer Update README.md

- R typo exercise corrected
- data-raw geodata upload
- output knitting everything
- plots last knit for tonight
- .Rbuildignore Initial commit
- .Rhistory updates cont
- .gitattributes Initial commit
- .gitignore Initial commit
- Course4\_MoveQ.Rproj Initial commit
- DESCRIPTION Initial commit
- NAMESPACE Initial commit

Go to file Add file · Code · Clone · HTTPS · SSH · GitHub CLI · https://github.com/stephkramer/Course4\_ · Open with GitHub Desktop · Download ZIP · 6 days ago

About

Introduction to movement data analysis

Readme · 0 stars · 2 watching · 0 forks

Packages

No packages published · Publish your first package

Contributors

# R-Studio/ File/ Rmarkdown -> rmd-file

The screenshot shows the RStudio interface with several windows open:

- Code Editor:** Shows R code for installing spatial packages and running a vignette.
- Console:** Shows the output of the R CMD INSTALL command for the dismo package.
- R Markdown Viewer:** Displays the generated R Markdown file content.
- Environment:** Shows an empty global environment.
- Help:** Shows the documentation for the dismo package, including its description and author information.

A large callout bubble with a green arrow points from the text "To follow what I show you at the screen, you can run the green arrow" to the green arrow icon in the R Markdown viewer window.

**To follow what I show you at the screen, you can run the green arrow**

**Or copy-paste line by line by using CTR+R**



END of day 1