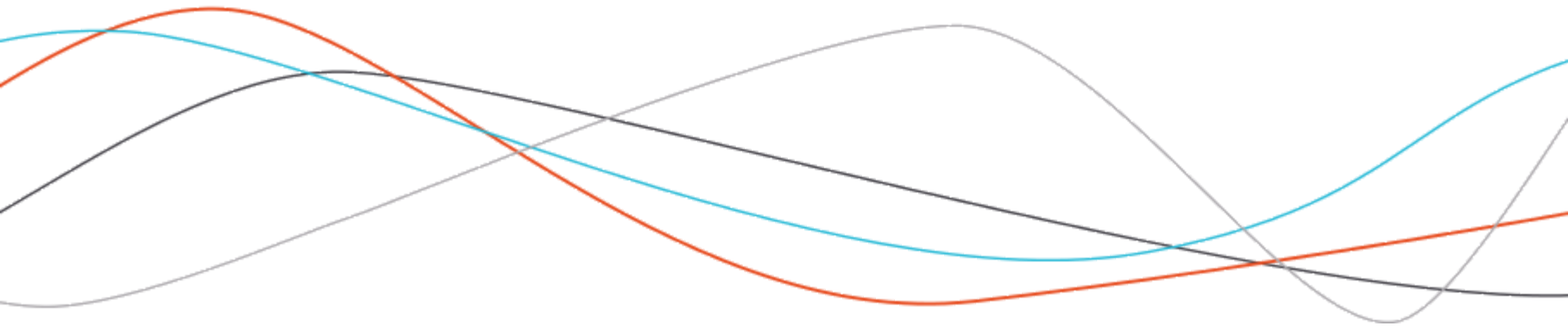
THINKR

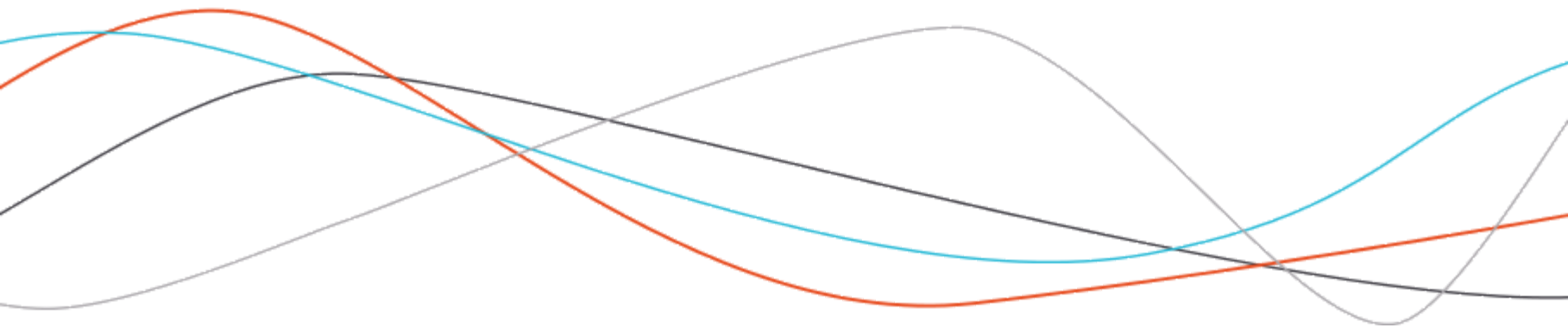
R -broaden data knowledge

October 10, 2018



Perform statistical tests

Test effects, investigate relationships between variables



What is a statistical test?

A statistical test is a process of rejecting or not rejecting (rarely accepting) a statistical hypothesis, called a null hypothesis, based on a dataset (sample). These are inferential statistics: based on calculations made on observed data, we draw conclusions about the population, linking them with the risk of error. (wikipedia)"

A statistical test only makes sense in the context of a **sampling** and a **scientific approach**.

If `p-value > chosen threshold` I can't "ACCEPT H0" (I just can't reject it)

Just because I can't demonstrate that a difference exists, does not mean that it doesn't exist

Just because I have demonstrated something "statistically", does not make it relevant

The different possible hypothesis tests

Sample A compared to sample B.

Two-sided test (test of difference)

```
H0: A = B.
H1: A ≠ B.
```

Unilateral test (superiority or inferiority test)

```
H0 : A = B.
H1 : A > B or H1 : A < B.
```

How to choose?

- by default we work bilaterally.
- you must make the choice BEFORE you do the test.
- one-sided tests are more powerful and detect an effect more easily. They can be "poorly perceived".
- one-sided tests do not allow certain differences to be detected.

Example

Sample A: `1,2,3,3,4,5,6,7` and Sample B: `3,4,5,6,7,8`.

Is the average of the values in the A sample different from the average of the values in the B sample? Is the 5.5 average of sample B really different from the 4 average of sample A?

```
H0: average of A is equal to average of B (A = B)
H1: average of A is different from average of B (A≠B)
```

The test gives us a critical probability (p): "chance of making a mistake by rejecting H0, i.e. by accepting H1" (which is rigourously wrong)

"If H0 is true then I only have "p" chance that the real difference between A and B is at least as big as the difference observed" (**correct but not very intuitive**)

We compare "p" to the x% threshold for making a decision (1%, 5% or 8%... it depends, but to be defined BEFORE the test).

Type I and type II risks

		Réalité	
		H0 vrai	H0 fausse
Décision	H0 acceptée	1- α	β
	H0 rejetée	α	1-β

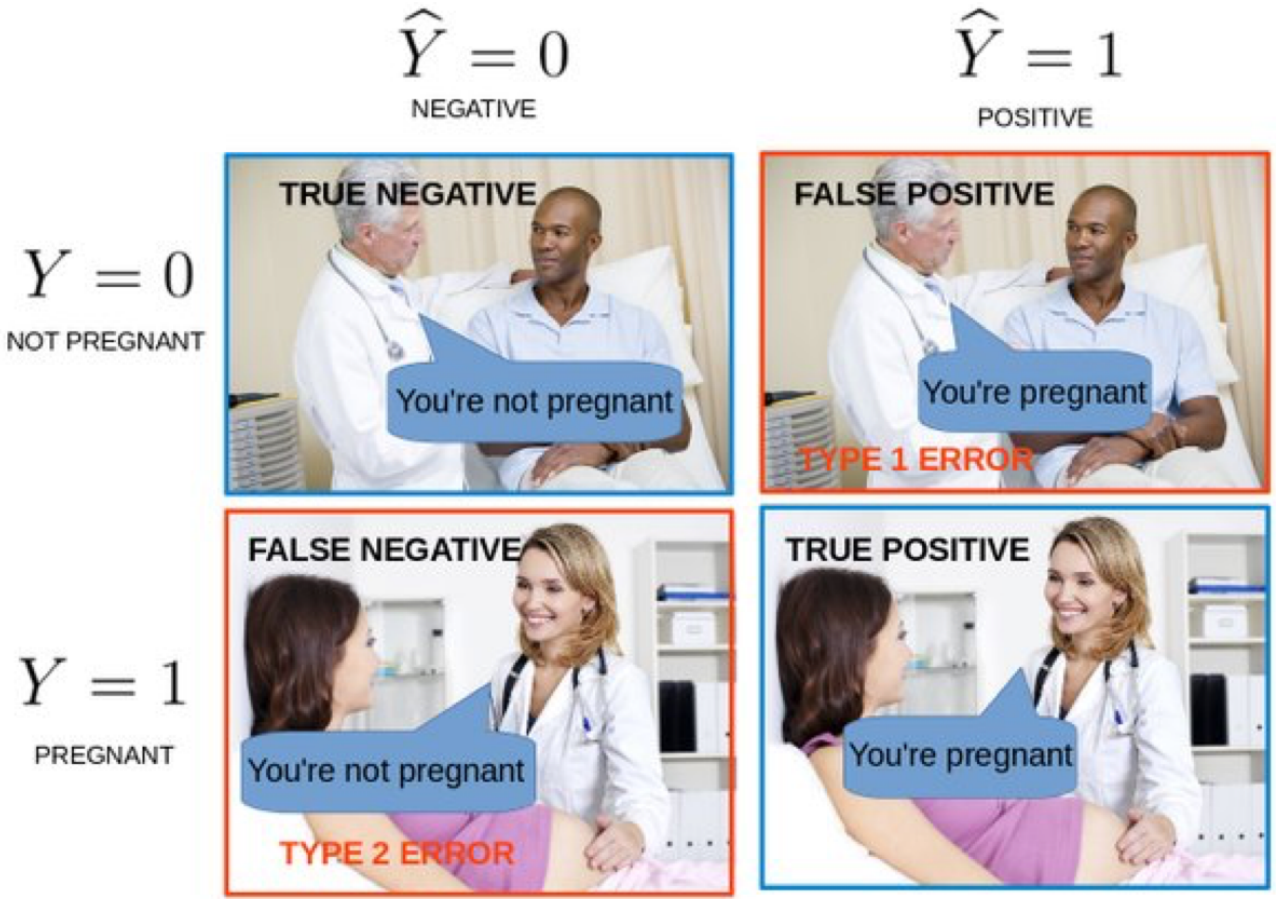
α : Type I risk

β : Type II risk

1- β : power of the test.

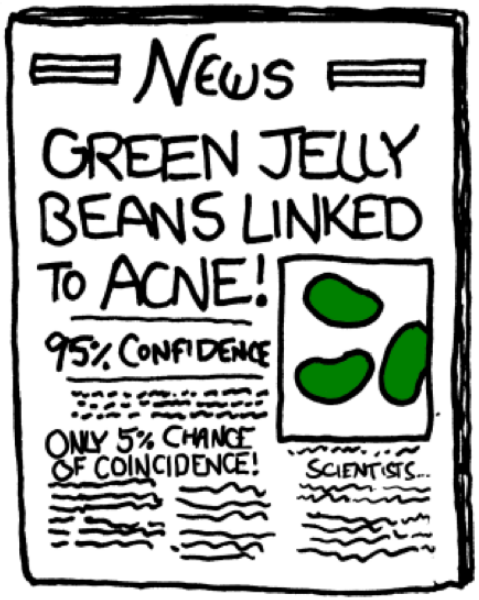
- α and β are user-defined. β depends on α, the size of the effect you want to detect and the number of individuals.
- The number of individuals to be interviewed does not depend on the size of the population.
- If I don't give myself the necessary experimental means I won't detect an effect if it exists.

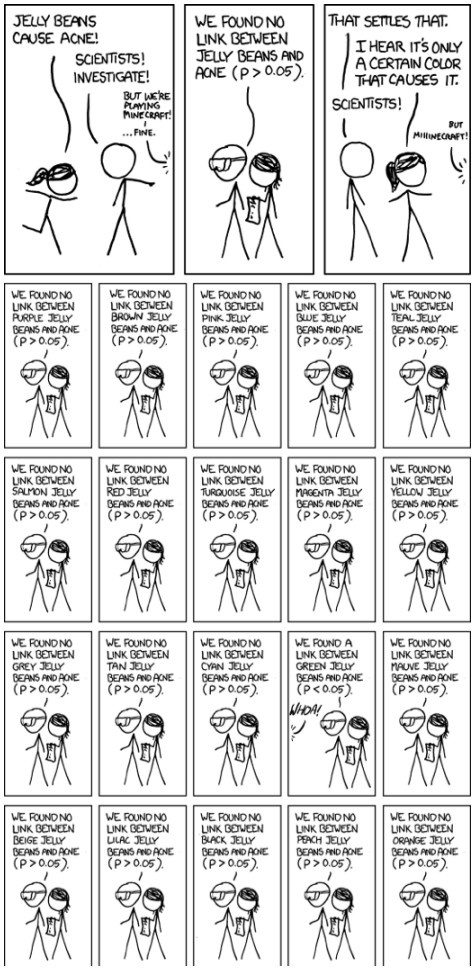
Type I and type II risks



About multiple tests

Risk control methods: Bonferroni, Holm, HSD, LSD....





Parametric or non-parametric tests?

Parametric statistical tests require that a certain number of hypotheses be respected (data normality, homo-sedasticity, etc.).

When the assumptions are not met, there is a non-parametric "equivalent" to the test, which does not require these assumptions.

Parametric or non-parametric tests?

Parametric statistical tests require that a certain number of hypotheses be respected (data normality, homo-sedasticity, etc.).

When the assumptions are not met, there is a non-parametric "equivalent" to the test, which does not require these assumptions.

BUT.

Non-parametric tests do not really do the same thing as parametric tests: in fact they answer another question!

Non-parametric tests are less powerful.

Parametric tests are generally very robust even outside the assumptions.

Parametric or non-parametric tests?

Parametric statistical tests require that a certain number of hypotheses be respected (data normality, homo-sedasticity, etc.).

When the assumptions are not met, there is a non-parametric "equivalent" to the test, which does not require these assumptions.

BUT.

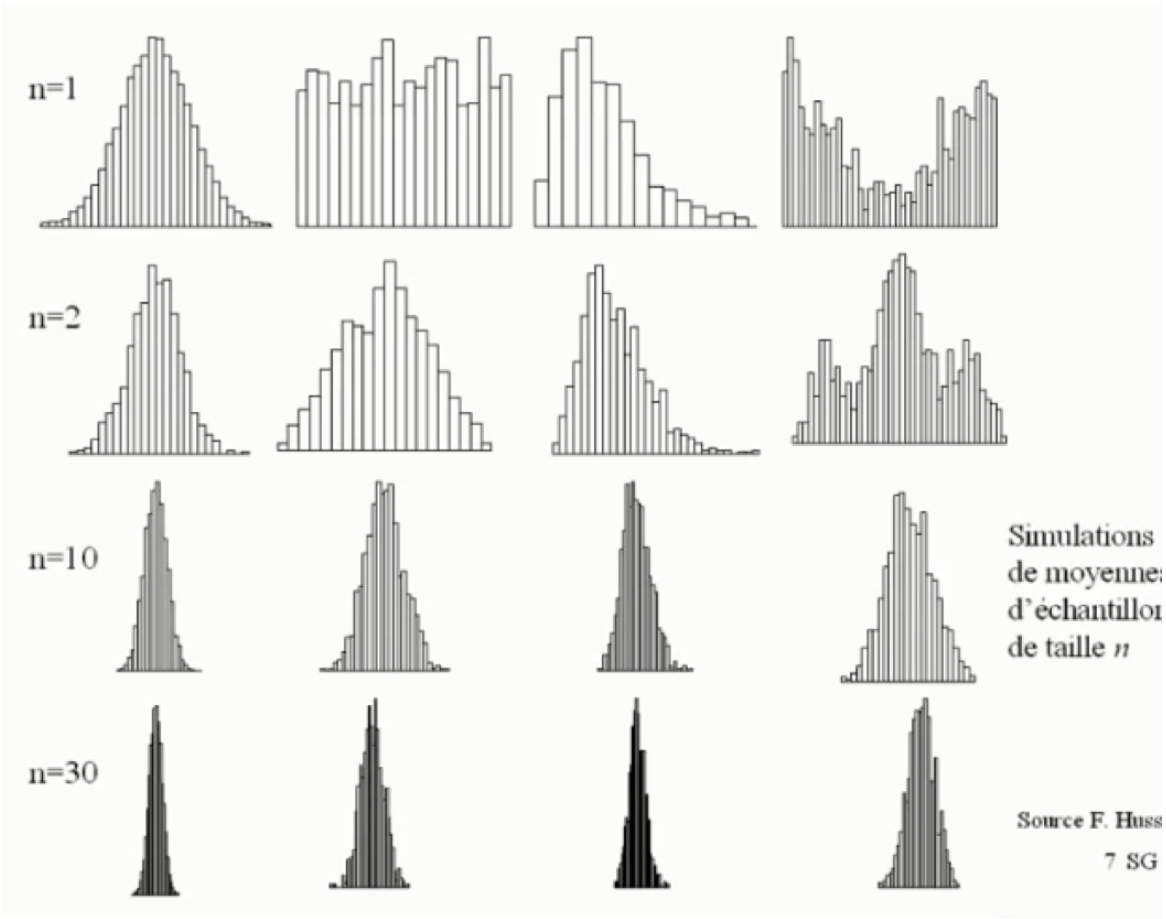
Non-parametric tests do not really do the same thing as parametric tests: in fact they answer another question!

Non-parametric tests are less powerful.

Parametric tests are generally very robust even outside the assumptions.

It is better to do a parametric test that is a little less correct than a non-parametric test.

About sample sizes



The different tests presented

- Normality Test - Shapiro-Wilk Test
- Comparison of 2 variances - Fischer-Snedecor test
- Average comparison test - Student test
- Comparison of averages/non-parametric - Mann-withney-wilcoxon
- Correlation measurement
- Comparison of a proportion to a theoretical value
- Comparison of two independent / matched proportions (Mac Nemar)
- Comparison of observed numbers of employees - Chi2 test
- ANOVA at 1 factor and +

The data set

```
library(readr)
tracks <- read_csv("data/tracks.csv")
# Transform the duration column into seconds
tracks$duration <- tracks$duration / 1000

group_a <- tracks[tracks$explicit,]
group_b <- tracks[!tracks$explicit,]
```

Normality Test (Shapiro-Wilk)

Objective

Does my variable fit well with a normal distribution?

Condition :

- Random sampling of each individual.
- Avoid if identical repeated values (see Chi2 of conformity).

Function :

```
shapiro.test()
```

Hypotheses

H0 : Our data follow a normal distribution

H1: Our data do not follow a normal distribution

Normality Test (Shapiro-Wilk)

Application :

```
A <- rnorm(100,mean = 5, sd = 3)
B <- runif(100,min = 2, max = 5)
shapiro.test(A)
```

```
#>
#>     Shapiro-Wilk normality test
#>
#> data:  A
#> W = 0.99465, p-value = 0.9651
```

```
shapiro.test(B)
```

```
#>
#>     Shapiro-Wilk normality test
#>
#> data:  B
#> W = 0.96454, p-value = 0.008587
```

Test of Normality (Shapiro-Wilk)

Question:

Does the duration column follow a normal distribution?

Comparison of 2 variances - Fischer-Snedecor test.

Objective :

Test if the variances of two populations are different.

Condition :

- Random sampling of each individual.
- Normal law in the 2 samples (not very robust).

Function :

```
var.test()
# Non-parametric alternative: Ansari-Bradley test.
ansari.test()
```

Hypotheses

H0: var(1) == var(2)

H1: var(1) != var(2)

Comparison of 2 variances - Fischer-Snedecor test.

Function :

```
A <- rnorm(100,mean = 5,sd = 3)
B <- rnorm(100,mean = 2,sd = 2)
var.test(A,B)
```

```
#>
#>     F test to compare two variances
#>
#> data:  A and B
#> F = 2.0814, num df = 99, denom df = 99, p-value = 0.0003204
#> alternative hypothesis: true ratio of variances is not equal to 1
#> 95 percent confidence interval:
#>  1.400467 3.093479
#> sample estimates:
#> ratio of variances
#>           2.081422
```

Comparison of 2 variances - Fischer-Snedecor test.

Application :

Compare the variance of the popularity variables of the two groups.

Average Comparison Test - Student Test

Objective :

Compare the averages of two populations, or to a target value.

Condition :

- Random sampling
- Normal law in the 2 samples (robust)

Function :

```
t.test()
# x = 1st sample, y = second sample, mu = target value,
# var.equal = TRUE/FALSE (put true if variance equal)
# paired = TRUE/FALSE (set TRUE if paired data)
# alternative = "two.sided" "less" or "greater"
```

Hypotheses

H0 : mean(1) == mean(2)

H1 : mean(1) != mean(2)

Question

Comparison with a target value:

Is the average length of the petals of the "setosa" flowers different from 0,5cm?

- Test normality

```

```

- Test difference

```

```

Question

Comparison of two independent samples:

Is the average length of the petals of "setosa" flowers different from that of "versicolor" flowers?

- Test Normality

```

```

- Test difference

```

```

Question

Comparison of two paired samples:

Paired means that the SAME individuals are found in the 2 compared samples (example: before and after a treatment).

Did the math averages of a group of students change between the first and third term?

```
note1 <- c(17,14,8,16,11,17,4,13)

note3 <- c(17,14,10,16,14,18,8,14)
```

Have they increased?

Question

Are the notes normal?

Question

- Are the averages different?

- Have they increased?

Comparison of averages / non-parametric (~comparison of medians)

Wilcoxon test (Mann-withney-wilcoxon)

Objective :

Allows to know if the medians of two populations are different or not, or if the observed median is different from a target value.

Condition :

- Random sampling
- Law of the same form in both samples

Function :

```
wilcox.test()
```

Hypotheses

H0 : mean(1) == mean(2)

H1 : mean(1) != mean(2)

Comparison of averages / non-parametric (~comparison of medians)

Application :

```
wilcox.test(x =
group_b$popularity, mu = 5)
```

```
#>
#>     Wilcoxon signed rank test with
continuity correction
#>
#> data:  group_b$popularity
#> V = 1873200, p-value < 2.2e-16
#> alternative hypothesis: true
location is not equal to 5
```

Application :

```
wilcox.test(x =
group_a$popularity, y =
group_b$popularity)
```

```
#>
#>     Wilcoxon rank sum test with
continuity correction
#>
#> data:  group_a$popularity and
group_b$popularity
#> W = 258580, p-value = 6.544e-05
#> alternative hypothesis: true
location shift is not equal to 0
```

Correlation measurement

Objective :

Allows to know to what extent two quantitative variables are correlated (linear relationship for Pearson and monotonous for Kendall and Spearman).

Condition :

- Random sampling
- Must follow a normal law (for Pearson)
- Must follow a monotonous relationship (for Kendall and Spearman)
- No missing data (individual deleted otherwise)
- A priori linear relationship (for Pearson)

Function :

```
cor.test()
# method: "pearson", "kendall", "spearman"
```

Hypotheses

H0 : correlation is 0

H1: correlation is different from 0

Correlation measurement

Application :

```
a <- sample(seq(1,10,0.1),10)
b <- sample(seq(1,10,0.1),10)
cor.test(a,b)
```

```
#>
#>     Pearson's product-moment correlation
#>
#> data:  a and b
#> t = -0.088535, df = 8, p-value = 0.9316
#> alternative hypothesis: true correlation is not equal to 0
#> 95 percent confidence interval:
#>  -0.6481452  0.6103632
#> sample estimates:
#>         cor
#> -0.03128656
```

Correlation measurement

Application :

Is duration related to popularity?

Comparison of a proportion to a theoretical value

Objective :

Allows to know if a proportion measured in a sample is different from a target value.

We compare `Pobs` to `Ptheo`, sample size n.

Condition :

- Random sampling
- n Pobs ≥ 5 and n (1-Pobs) ≥ 5

Function :

```
binom.test()
# x : number of successes; n : number of attempts
# p : theoretical probability
```

Hypotheses

H0 : proportion == x

H1: proportion != x or <= x or >= x

Application :

```
# 87 "yes" out of 100 respondents
is it different from 80%?
binom.test(87,100,0.8)
```

```
#>
#>     Exact binomial test
#>
#> data:  87 and 100
#> number of successes = 87, number of
trials = 100, p-value =
#> 0.08106
#> alternative hypothesis: true
probability of success is not equal to
0.8
#> 95 percent confidence interval:
#>  0.7879593 0.9289270
#> sample estimates:
#> probability of success
#>                   0.87
```

Application :

```
# Is the proportion of men 0.8?
binom.test(2162, 4367,
           0.8)
```

```
#>
#>     Exact binomial test
#>
#> data:  2162 and 4367
#> number of successes = 2162, number
of trials = 4367, p-value <
#> 2.2e-16
#> alternative hypothesis: true
probability of success is not equal to
0.8
#> 95 percent confidence interval:
#>  0.4801409 0.5100191
#> sample estimates:
#> probability of success
#>              0.4950767
```

Comparison of two independent proportions

Objective :

Allows to know if two measured proportions are identical or not

Condition :

- Random sampling in each sample
- Contingency table must show more than 5 individuals in each cell
- Independent samples

Function :

```
prop.test()
```

Hypotheses

H0 : proportions are ==

H1: proportions are !=

Application :

```
# Is the proportion of female smokers the same as that of male smokers?
mat <- matrix(c(15,10,15,20),2)
dimnames(mat) <- list(c("man", "woman"), c("yes", "no"))
prop.test(mat)
```

```
#>
#>     2-sample test for equality of proportions with continuity
#>     correction
#>
#> data:  mat
#> X-squared = 1.0971, df = 1, p-value = 0.2949
#> alternative hypothesis: two.sided
#> 95 percent confidence interval:
#>  -0.1125679  0.4459012
#> sample estimates:
#>    prop 1    prop 2
#> 0.5000000 0.3333333
```

Application :

Is the proportion of "explicit" songs != "not explicit" songs?

Comparison of two paired proportions

Objective :

Allows to know if two measured paired proportions are identical or not.

Condition :

- Random sampling in each sample
- Contingency table must show more than 5 individuals in each cell
- Individuals must be able to move from one state to another

Function :

```
mcnemar.test()
```

Application :

```
# Has the proportion of smokers
varied over time?
mat <- matrix(c(20,2,10,28),2)
dimnames(mat) <- list("before" =
c("smoker", "non-smoker"),
   "after" = c("smoker", "non-
smoker"))
mcnemar.test(mat)
```

Comparison of observed numbers of individuals

Objective :

Allows to know if there is a link between 2 qualitative variables.

Condition :

- Random sampling in each sample
- Contingency table must show more than 5 individuals in each cell (see `fisher.test()` otherwise)

Function :

```
chisq.test()
```

Hypothesess

H0 = the two variables are independent

H1 = the two variables are not independent

Comparison of observed numbers of individuals

Application :

```
# Does age influence the probability to smoke?
mat <- matrix(c(15,10,15,20,10,10),3)
dimnames(mat) <- list(c("15-30","31-45","46-70"),
                      c("yes","no"))
chisq.test(mat)
```

```
#>
#>     Pearson's Chi-squared test
#>
#> data:  mat
#> X-squared = 1.7143, df = 2, p-value = 0.4244
```

Questions

import the file `ozone.csv` into an object called ozone:

```

```

- Check the import by looking at the basic statistics by variables (`summary`)

```

```

- Build a contingency table between variables `ozone$pluie` and `ozone$vent` (function `table`)

```

```

- Test if there is a relationship between wind direction and the fact that it is raining:

```

```

Anova

Objectives :

Test if the dispersion of data is due to one or more known variables.
Explain a quantitative variable from qualitative variables.

Conditions :

- Random sampling
- Normal law in sub-populations (tested *a posteriori*)
- If not normal, see `kruskal.test`
- Homogeneity of variance in sub-populations

Function :

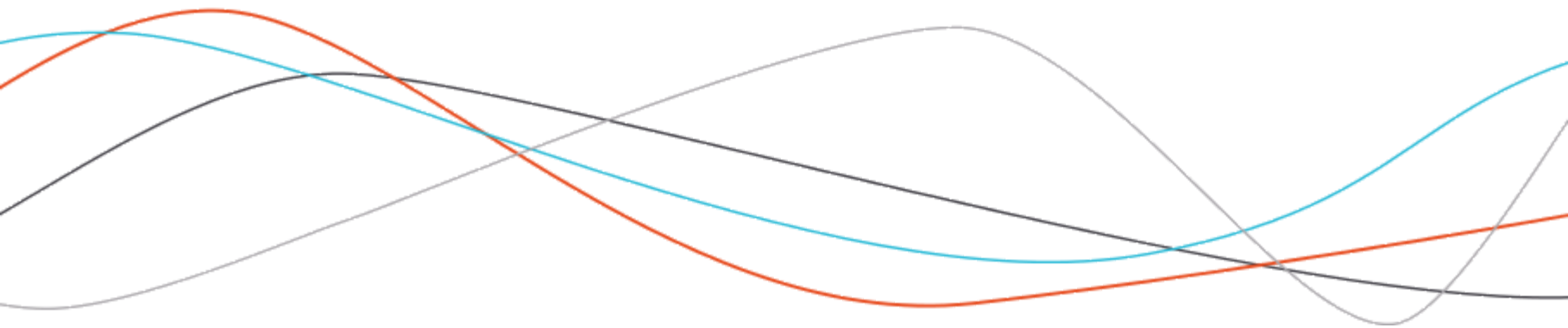
```
aov, AovSum (package{FactomineR}),
anova, lm
```

Application :

```
# Use a formula as a parameter:
ToExplain ~ factor1 + factor2 +
factor3
# or
ToExplain ~ factor1 + factor2 +
factor1:factor2
```

Multivariates analyses

Test effects, estimate relation between variables



ANOVA with 1 factor and +

Objectives :

Test if the dispersion of the data is due to one or more known variables.
Explain a quantitative variable from qualitative variables.

Conditions :

- Random sampling
- Normal law in sub-populations (tested *a posteriori*)
- If not normal, see `kruskal.test`
- Homogeneity of variance in sub-populations

Function :

```
aov, AovSum (package{FactomineR}),
anova, lm
```

Application :

```
# Use a formula as a parameter:
ToExplain ~ factor1 + factor2 +
factor3
# or
ToExplain ~ factor1 + factor2 +
factor1:factor2
```

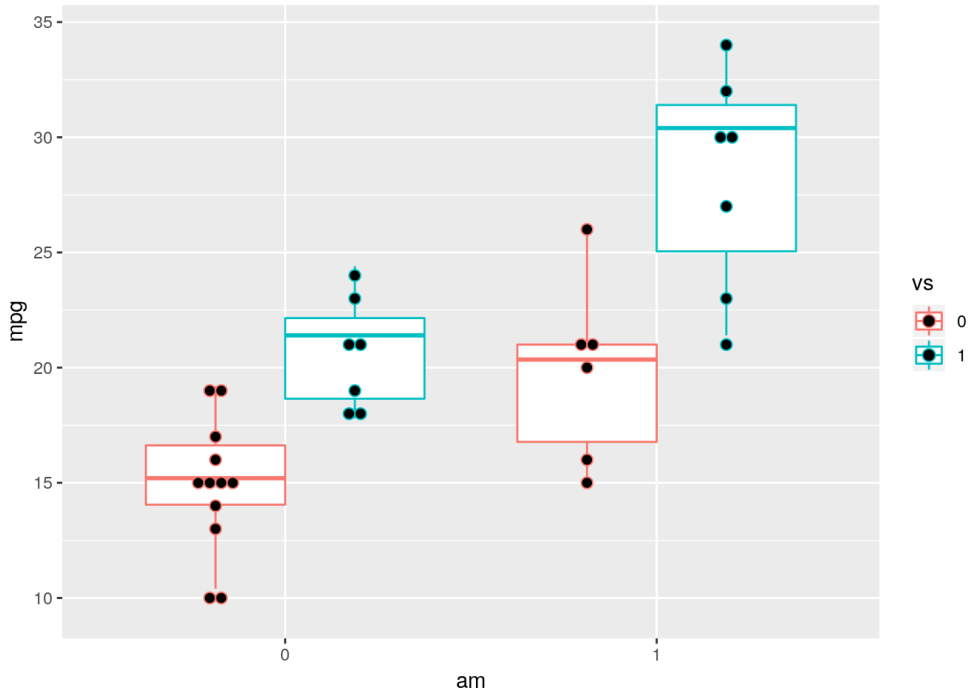
ANOVA with 1 factor and +

Example

`mtcars` : Distance covered with a petrol galon (`mpg`) ~ transmission type (`am`=0 automatic, `am`=1 manual) + engine type (`vs`=0 V-shaped, `vs`=1 straight)

Exploration

Visual verification of hypotheses



ANOVA with 1 factor and +

Estimate

```
AovSum(mpg ~ am * vs, data = mtcars_chr)
```

```
#> Ftest
#>                SS df     MS F value    Pr(>F)
#> am         283.72  1 283.72 23.5400 4.159e-05 ***
#> vs         382.48  1 382.48 31.7337 4.931e-06 ***
#> am:vs       16.01  1  16.01  1.3283    0.2589
#> Residuals 337.48 28  12.05
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Ttest
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 20.97857    0.63526 33.0238   <2e-16 ***
#> am          -3.08214    0.63526 -4.8518    4e-05 ***
#> vs          -3.57857    0.63526 -5.6333   <2e-16 ***
#> am : vs      0.73214    0.63526  1.1525   0.2589
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Linear models

Objectives :

Test if the average effect of one or more variables is different from zero
Explain a quantitative or qualitative variable from other variables

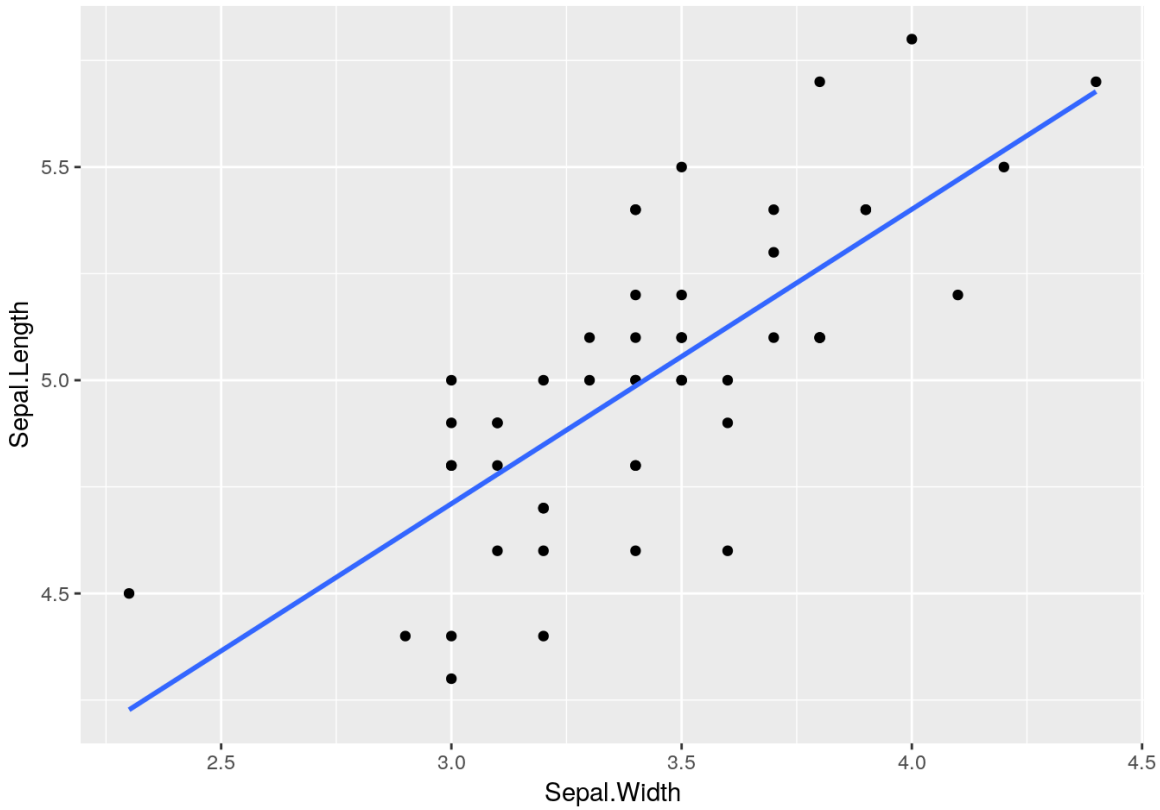
Conditions :

- Random sampling
- Normal law in sub-populations (tested *a posteriori*)
- Homogeneity of variance in sub-populations

Linear models

Exploration

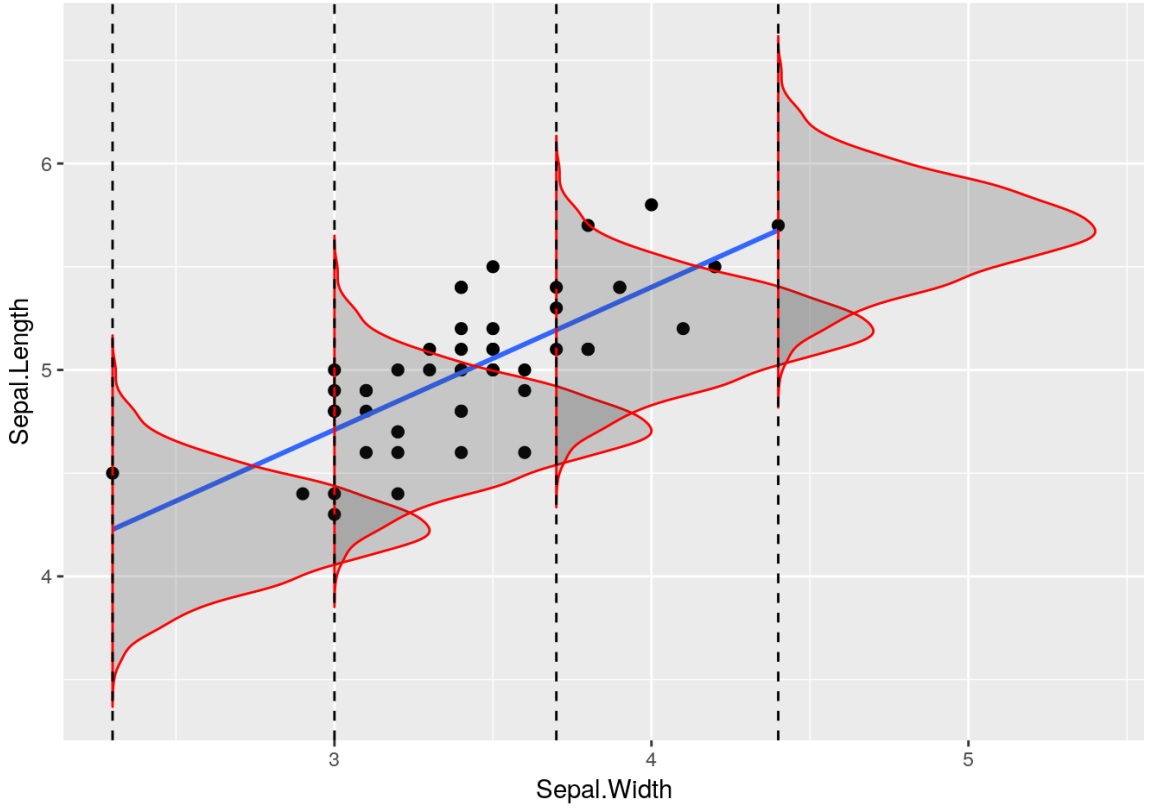
`iris[setosa]` : Sepal length (`Sepal.Length`) ~ Sepal width (`Sepal.Width`)



Linear models

Theoretical assumptions

- Gaussian distribution of **residuals**
- Homogeneity of the variance of **residuals**



Linear models

Model fitting

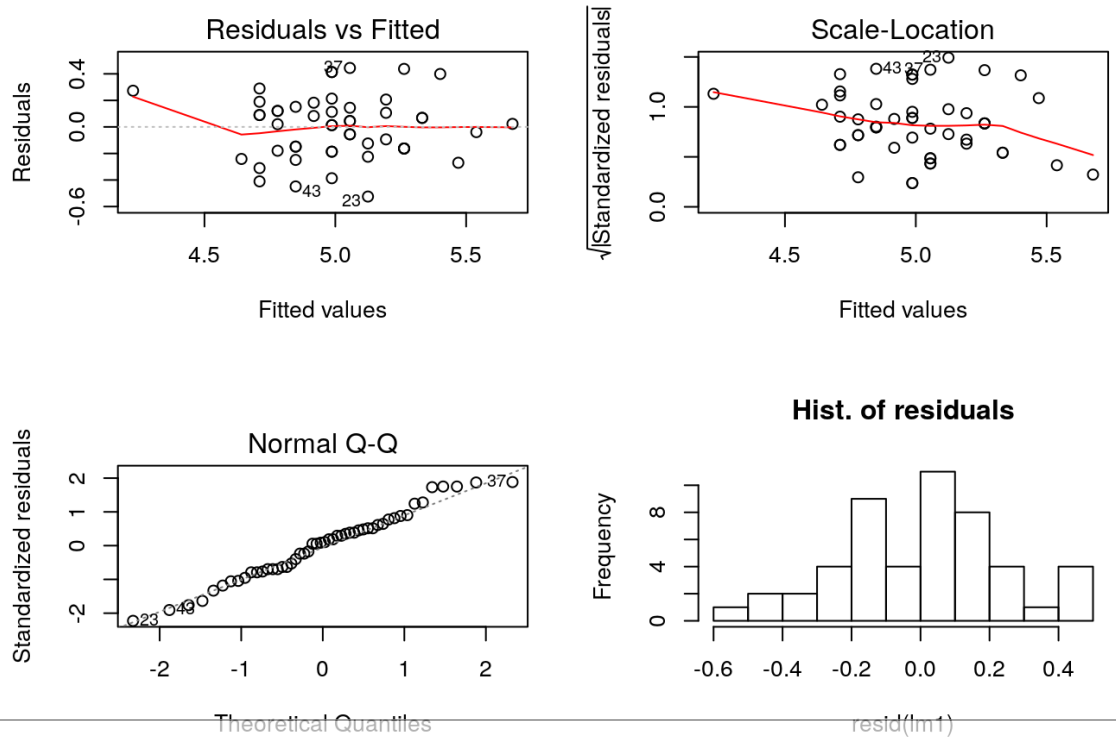
```
lm1 <- lm(Sepal.Length ~ Sepal.Width, data = iris_setosa)
summary(lm1)
```

```
#>
#> Call:
#> lm(formula = Sepal.Length ~ Sepal.Width, data = iris_setosa)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -0.52476 -0.16286  0.02166  0.13833  0.44428
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   2.6390     0.3100   8.513 3.74e-11 ***
#> Sepal.Width   0.6905     0.0899   7.681 6.71e-10 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.2385 on 48 degrees of freedom
#> Multiple R-squared:  0.5514,    Adjusted R-squared:  0.542
#> F-statistic: 58.99 on 1 and 48 DF,  p-value: 6.71e-10
```

Linear models

Checking assumptions *a posteriori*

```
par(mfcol = c(2, 2))
plot(lm1, which = 1:3)
hist(resid(lm1), main = "Hist. of residuals")
```

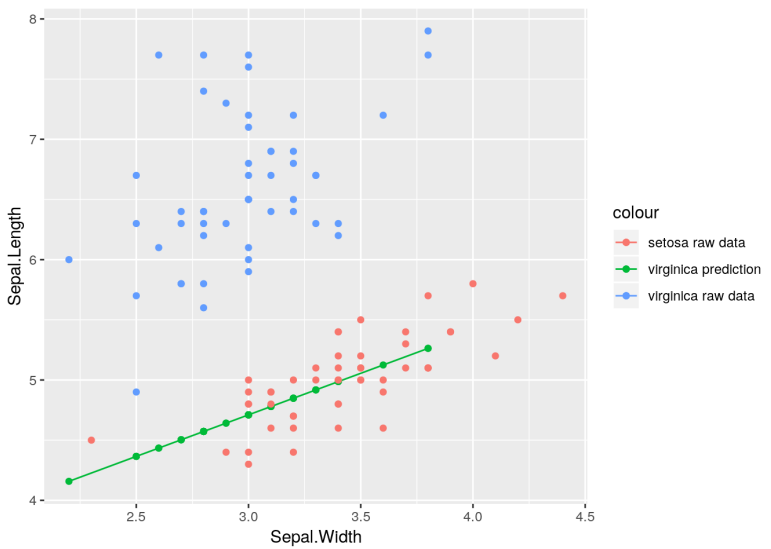


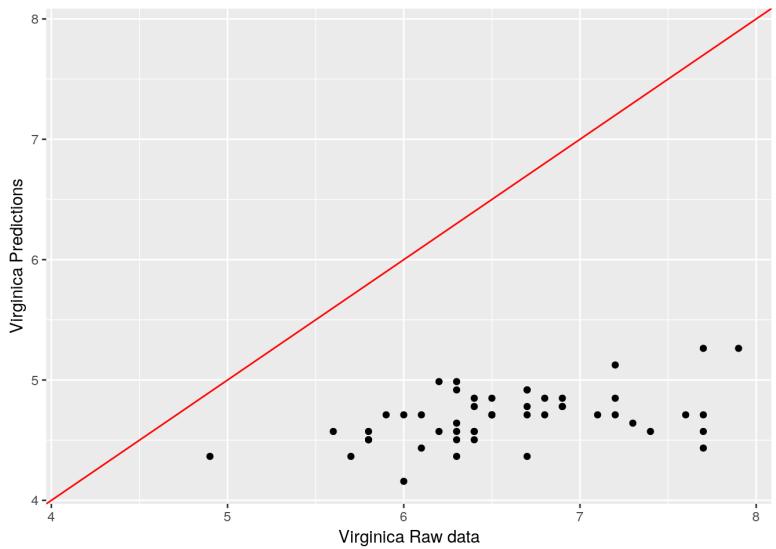
Linear models

Predictions

- `predict` allows to use the model to predict
  - The `newdata` parameter is used to predict on a new dataset

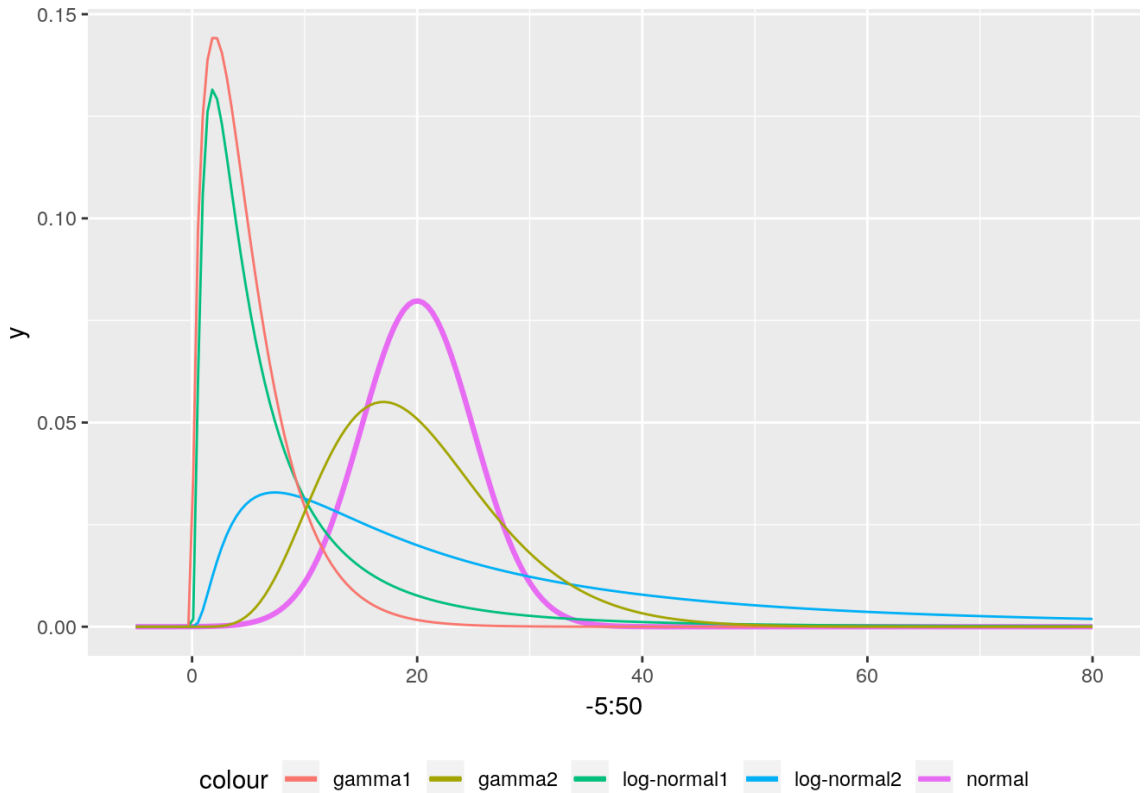
```
iris_virginica <- iris %>%
  filter(Species == "virginica") %>%
  mutate(pred_sepal_length = predict(lm1, newdata = .))
```



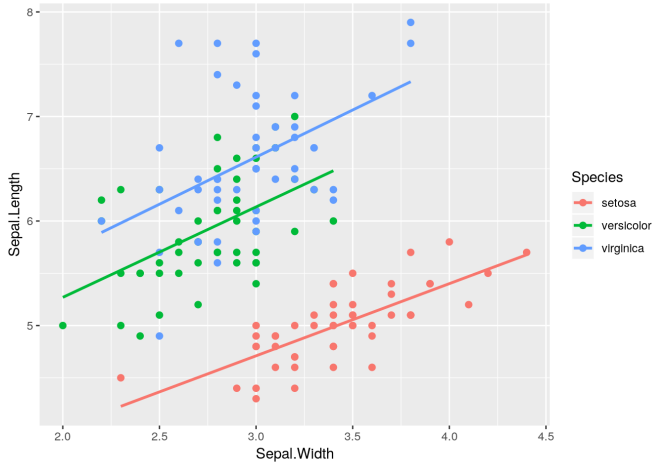


Generalized linear models (GLM)

- Distribution of residuals other than Gaussian
  - continuous or binomial data (logistic)



Generalized linear models (GLM)




```
lm1 <- glm(Sepal.Length ~ Sepal.Width * Species, data = iris, family =
"Gamma")
summary(lm1)$coefficients
```

```
#>                                    Estimate  Std. Error    t value
#> (Intercept)                     0.294135904 0.018800749 15.6449035
#> Sepal.Width                    -0.027375699 0.005390745 -5.0782772
#> Speciesversicolor              -0.056278495 0.024571417 -2.2904050
#> Speciesvirginica               -0.081385586 0.023672308 -3.4380081
#> Sepal.Width:Speciesversicolor   0.002450333 0.007788086  0.3146258
#> Sepal.Width:Speciesvirginica    0.006974010 0.007193782  0.9694497
#>                                    Pr(>|t|)
```

Model selection

- Akaike Criteria (AIC): the lower the better
- Criteria with 10-fold cross-validation[recommended]
  - RMSE for continuous data (Residual Mean Square Error)
  - AUC for binomial data (Area Under the Curve)

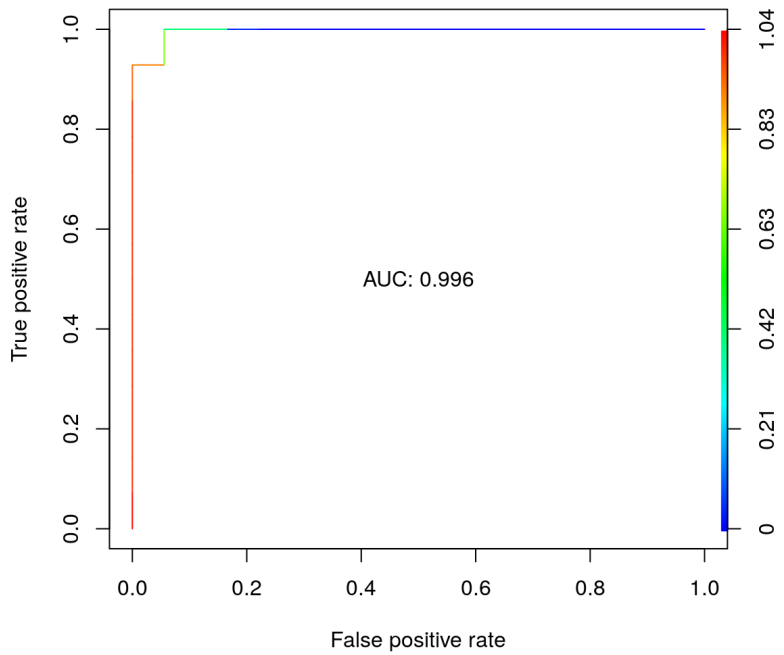
```
glm_complet <- glm(mpg ~ ., data = mtcars_chr)
res <- stepAIC(glm_complet, direction = c("both"))
```

```
#> Start:  AIC=169.06
#> mpg ~ cyl + disp + hp + drat + wt + qsec + vs + am + gear + carb
#>
#>        Df Deviance    AIC
#> - carb  5   146.56 163.50
#> - gear  2   129.82 165.62
#> - qsec  1   128.21 167.22
#> - am    1   128.25 167.24
#> - vs    1   128.73 167.35
#> - cyl   1   131.33 168.00
#> - drat  1   132.19 168.20
#> <none>      127.53 169.06
#> - disp  1   143.86 170.91
#> - hp    1   146.20 171.43
#> - wt    1   154.34 173.16
```

ROC and AUC curve

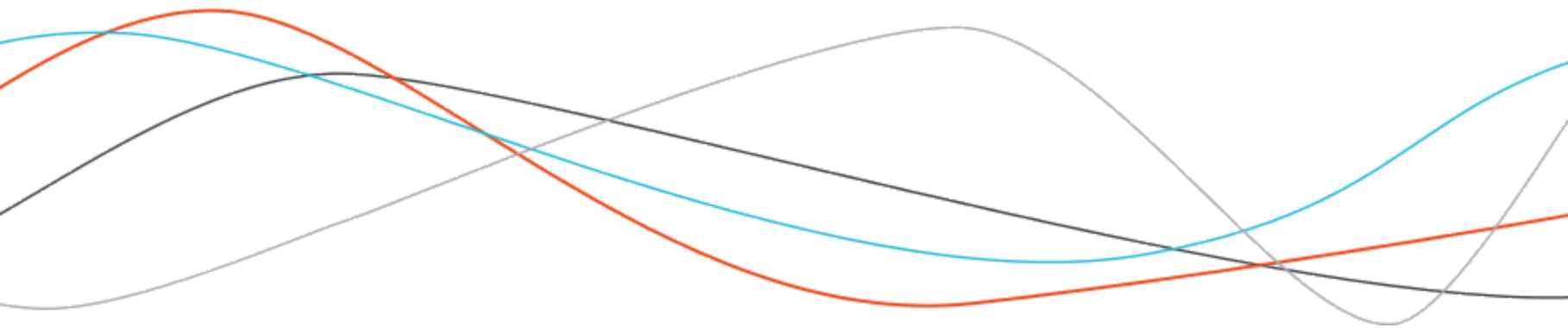
- Binomial data
  - Area Under the Curve: calculated from a ROC curve (Receiving Operating Characteristic) that compares the true positive rate (sensitivity) to the false positive rate (specificity) at various threshold values

```
mtcars_bin <- mutate(mtcars_chr,
  mpg_bin =
    (mpg > mean(mtcars_chr$mpg)))

glm_bin <- glm(mpg_bin ~ am + hp,
  data = mtcars_bin,
  family = "binomial")

pred <- predict(glm_bin,
  type = "response") %>%
  prediction(mtcars_bin$mpg_bin)

perf <- performance(pred,
  "tpr", "fpr")
```



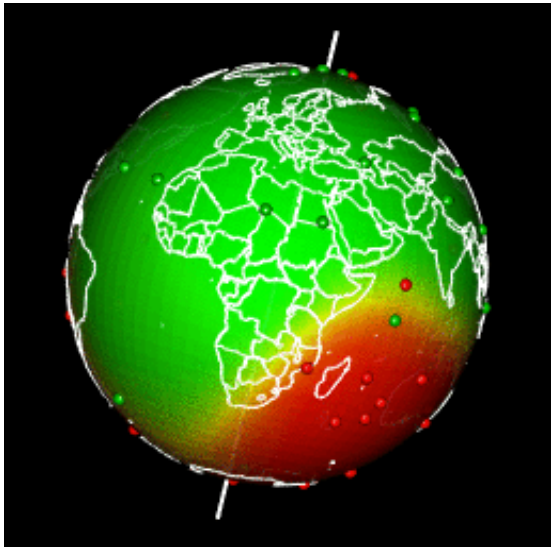
Basis of cartography

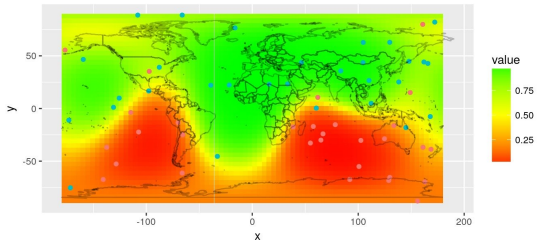
GIS and spatial analyses



What is a map ?

A 2-dimensional representation of all or part of our planet



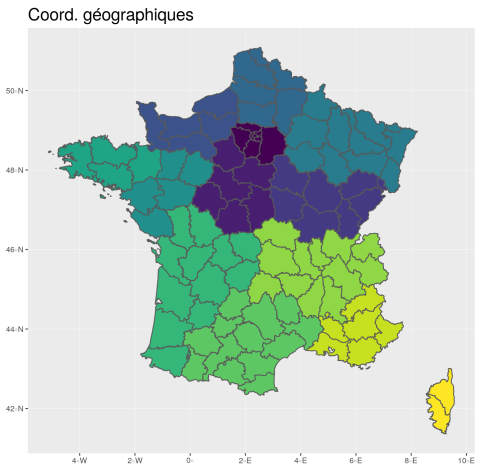


What is a projection?

The source of all your problems in cartography...

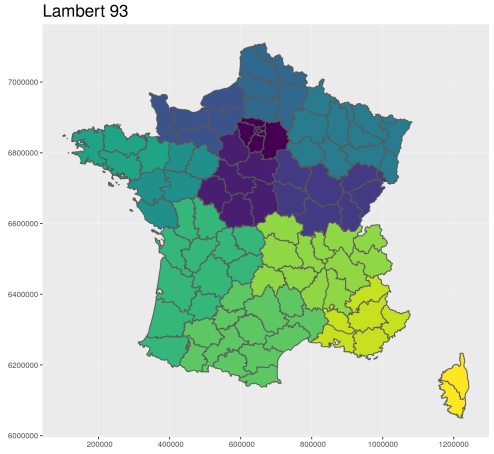
Geographic coordinates

- In degrees, minutes
- EPSG: 4326
- Use to share data



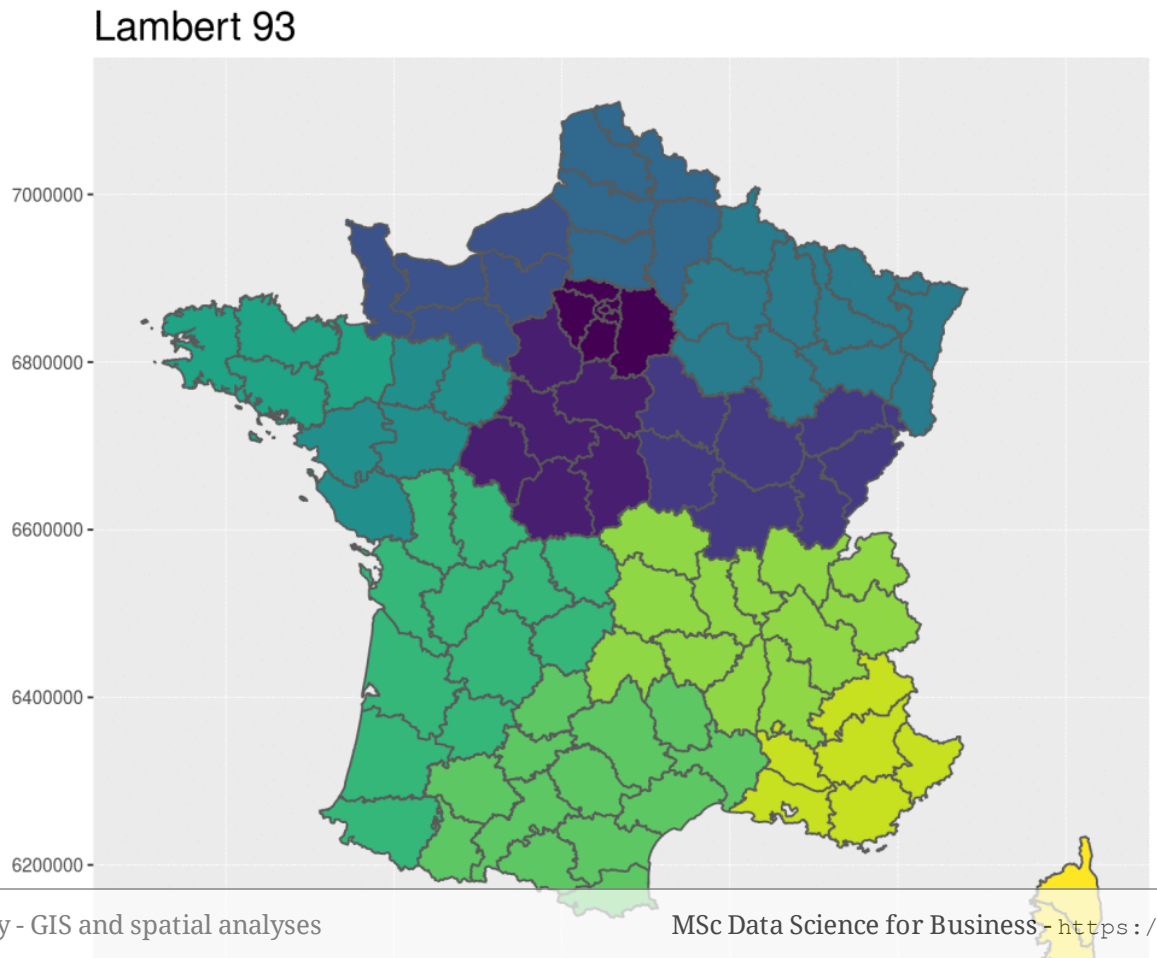
Projection Lambert 93

- Coordinates in metres
- EPSG: 2154
- A projection to display a map of Metropolitan France



What is a projection?

- Deformation during projection changes



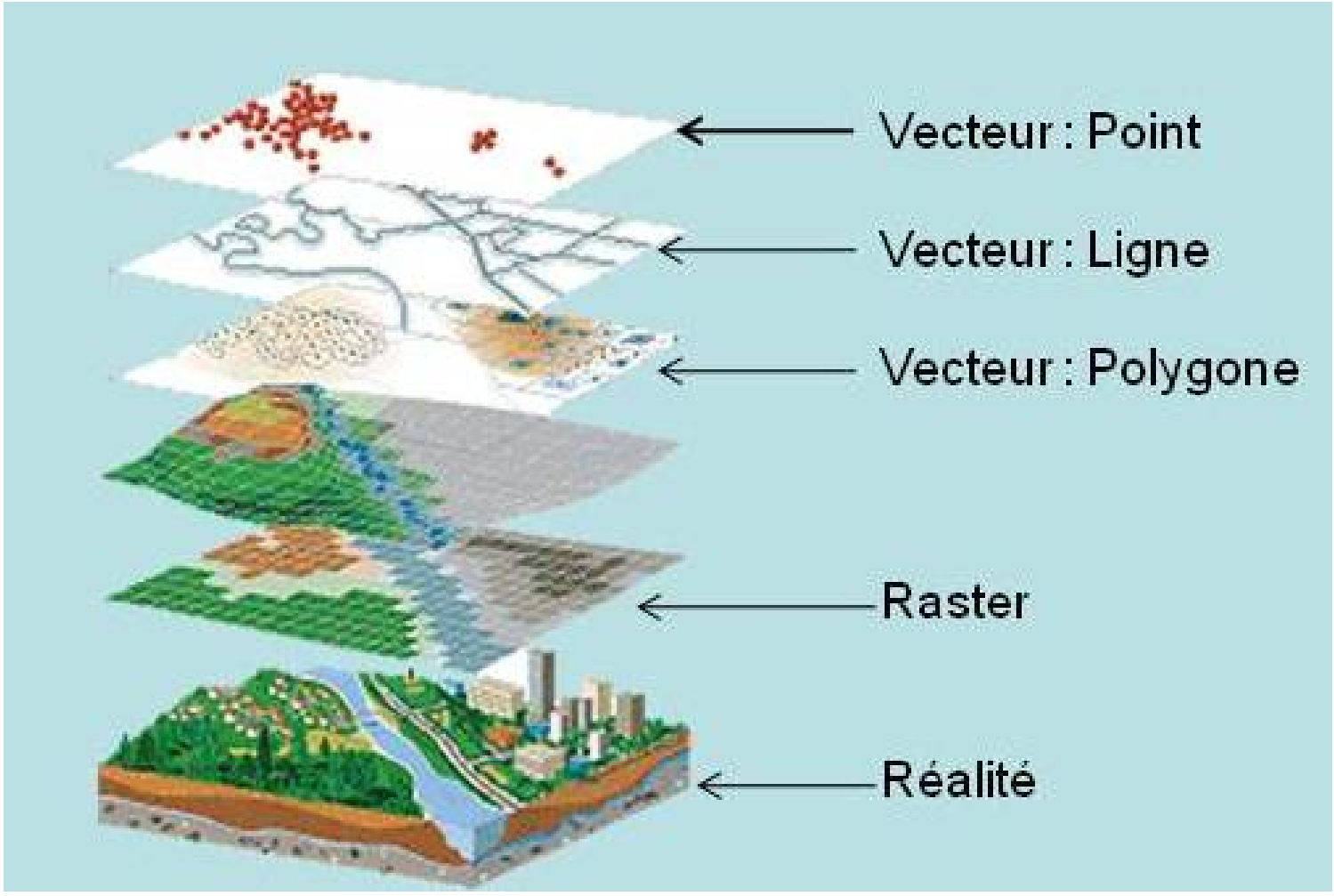
What is a projection?

- In R, you will talk about `CRS` (Coordinates Reference System)

Use the name of the CRS in the name of your spatial object! This will prevent you from making many mistakes...

```
france_l93
france_wgs84
world_merc
world_eck
```

Geographic objects (entities)



Vector objects

Geographic objects (entities):

- Points: entities without surface and which cannot be represented by lines
- Lines (arcs) :
  - Objects too narrow to be described by a surface (street, river -depending on the scale of representation-)
  - Objects with a length but no surface (coastline, departmental boundaries...)
- Polygons: Homogeneous surface objects (island, building,...)

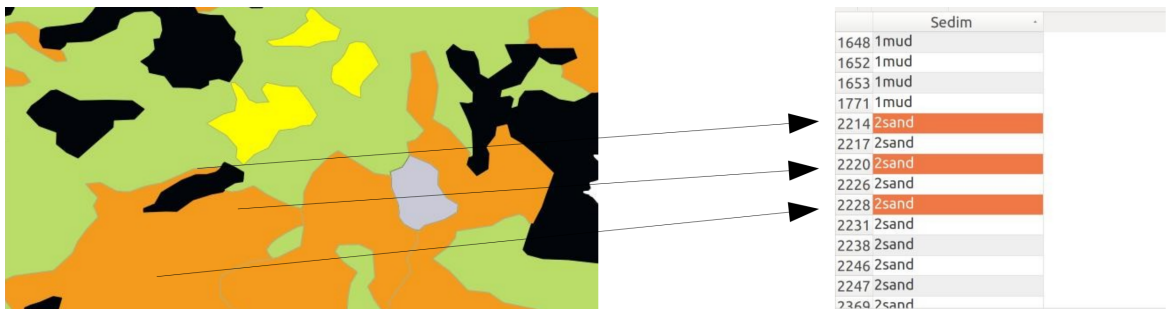
Storage is done in a single database or in several files

> One layer = one type of entity

Vector objects

Geographical entities are described by geometric shapes linked to an alphanumeric database

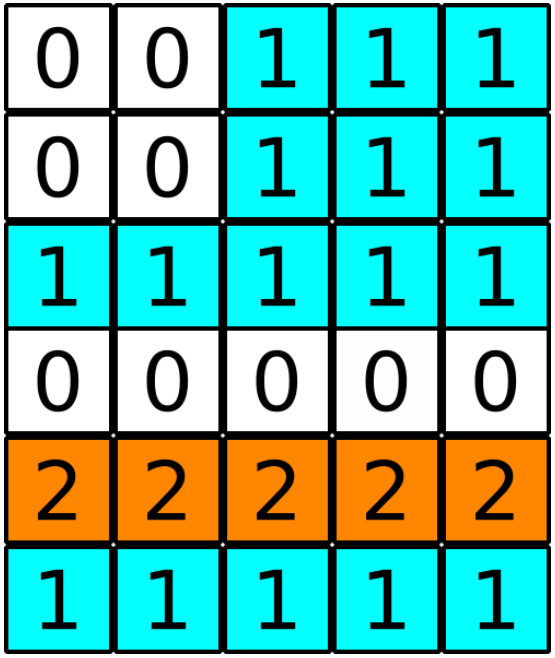
- Entity: geographical object represented on a map
- Attributes: database of information describing the entities (attribute table)



Raster objects (rasters)

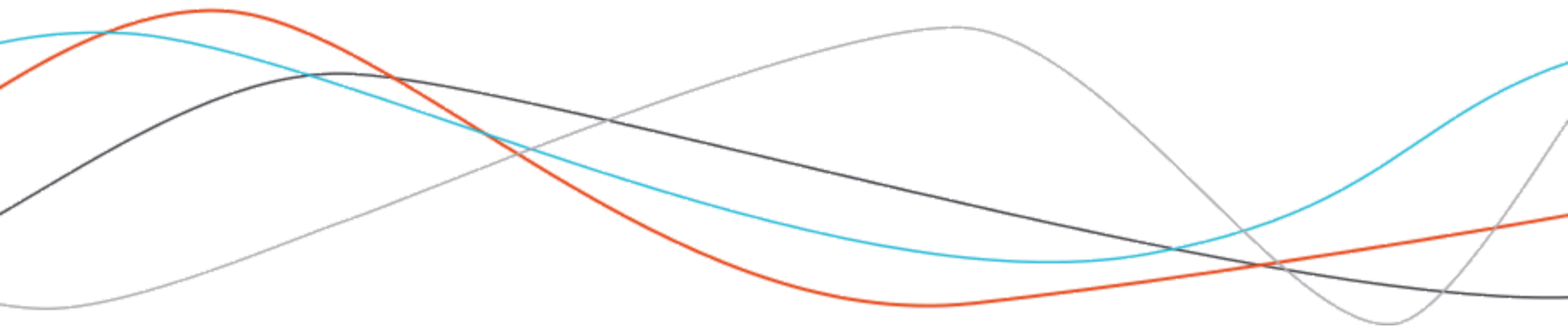
The storage is done in the form of a rectangular matrix of values

- Decomposition of regular meshes into grids
- Each mesh (=pixel) has a specific value related to its representation attribute (quality)
- A raster is defined by:
  - Its geographical extension
    - Number of rows, columns and geographical position (corner)
  - Its spatial resolution
    - Mesh size + Mesh size



Mapping vectors with R

Vectors



Mapping and spatial analysis with R

- Past: {sp} + {rgdal} + {rgeos} + {raster}
- Present: {sf} + {raster}
- Future: {sf} + {stars}

Reading vector layer files

Reading with `st_read`:

- All formats handled by GDAL (http://www.gdal.org/)
  - CSV, Mapinfo, Google Earth, GeoJSON, PostGIS, ...
- ESRI shapefile format
  - Most commonly shared
  - 4 files minimum (shp, shx, dbf, prj)

```
# dataWD <- "data"
departements_l93 <- st_read(
  dsn = file.path(dataWD, "carto-data-orig/departements"),
  layer = "DEPARTEMENT",
  quiet = TRUE) %>%
  st_transform(2154)
```

Preamble: Projections

- Get the coordinate system of a layer: `st_crs`

```
st_crs(departements_l93)
```

```
#> Coordinate Reference System:
#>   EPSG: 2154
#>   proj4string: "+proj=lcc +lat_1=49 +lat_2=44 +lat_0=46.5 +lon_0=3
+x_0=700000 +y_0=6600000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs"
```

- Project a layer into a new coordinate system: `st_transform`

```
# Change to geographical coordinates
departements_l93 %>%
st_transform(crs = 4326) %>%
st_crs()
```

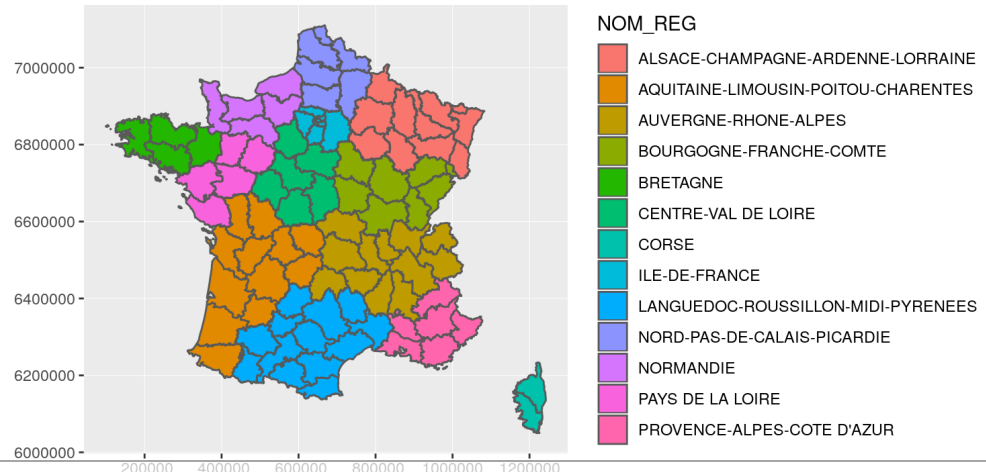
```
#> Coordinate Reference System:
#>   EPSG: 4326
#>   proj4string: "+proj=longlat +datum=WGS84 +no_defs"
```

Preamble: Displaying your maps with {gglot2}

The results of the following operations on {sf} objects can be viewed with {gpplot2}

- Display with `geom_sf` + `coord_sf`
  - Recognizes the type of entity (*point, line, polygon*)
  - Always set both to avoid projection problems

```
ggplot(departements_l93) +
  geom_sf(aes(fill = NOM_REG)) +
  coord_sf(crs = st_crs(departements_l93),
           datum = st_crs(departements_l93))
```



Vector layer file format

- With {sp} + {rgdal}

```
#> Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
#>   ..@ data       :'data.frame':    96 obs. of  2 variables:
#>   ..@ polygons   :List of 96
#>   ..@ plotOrder  : int [1:96] 57 12 27 69 76 21 63 18 65 26 ...
#>   ..@ bbox       : num [1:2, 1:2] 99217 6049646 1242417 7110480
#>   .. ..- attr(*, "dimnames")=List of 2
#>   ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot

#> Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
#>   ..@ data       :'data.frame':    96 obs. of  2 variables:
#>   .. ..$ CODE_DEPT: Factor w/ 96 levels "01","02","03",..: 40 43 77 90 69 27 10
56 62 68 ...
#>   .. ..$ NOM_DEPT : Factor w/ 96 levels "AIN","AISNE",..: 51 54 82 95 35 30 9 65
71 12 ...
#>   ..@ polygons   :List of 96
#>   .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
#>   .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
#>   .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
#>   .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
#>   .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
#>   .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
```

Vector layer file format

- With {sf} : read with `st_read`

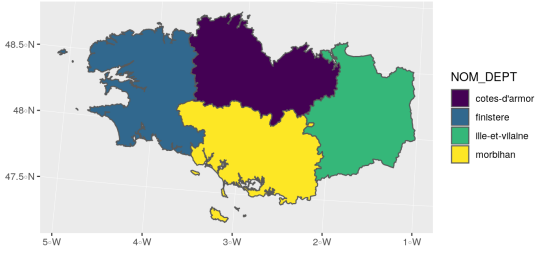
```
#> Simple feature collection with 96 features and 2 fields
#> geometry type:  MULTIPOLYGON
#> dimension:      XY
#> bbox:           xmin: 99217.1 ymin: 6049646 xmax: 1242417 ymax: 7110480
#> epsg (SRID):    2154
#> proj4string:    +proj=lcc +lat_1=49 +lat_2=44 +lat_0=46.5 +lon_0=3 +x_0=700000
+y_0=6600000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
#> First 10 features:
#>    CODE_DEPT       NOM_DEPT                       geometry
#> 1         39           JURA MULTIPOLYGON (((886244.2 66...
#> 2         42          LOIRE MULTIPOLYGON (((764370.3 65...
#> 3         76 SEINE-MARITIME MULTIPOLYGON (((511688.8 69...
#> 4         89          YONNE MULTIPOLYGON (((709449.1 67...
#> 5         68      HAUT-RHIN MULTIPOLYGON (((992779.1 67...
#> 6         28   EURE-ET-LOIR MULTIPOLYGON (((548948.9 68...
#> 7         10           AUBE MULTIPOLYGON (((740396.5 68...
#> 8         55          MEUSE MULTIPOLYGON (((846578.7 68...
#> 9         61           ORNE MULTIPOLYGON (((425026.6 68...
#> 10        67       BAS-RHIN MULTIPOLYGON (((998020.8 68...
```

Data manipulation with {dplyr}

Everything you know about {dplyr} works on the objects of {sf}

- `%>%`
- `select`, `mutate` for attributes (= columns)
- `filter`, `arrange` for entities (= rows)

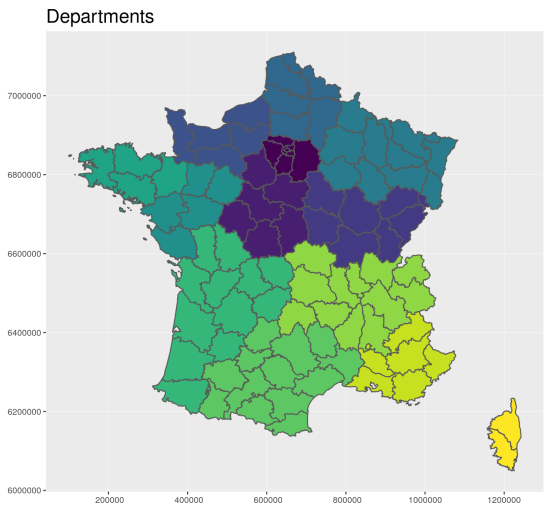
```
Bret_l93 <-
  departements_l93 %>%
  mutate_at(
    vars(NOM_DEPT, NOM_REG),
    tolower) %>%
  select(CODE_DEPT, NOM_DEPT,
NOM_REG) %>%
  filter(NOM_REG == "bretagne")
```

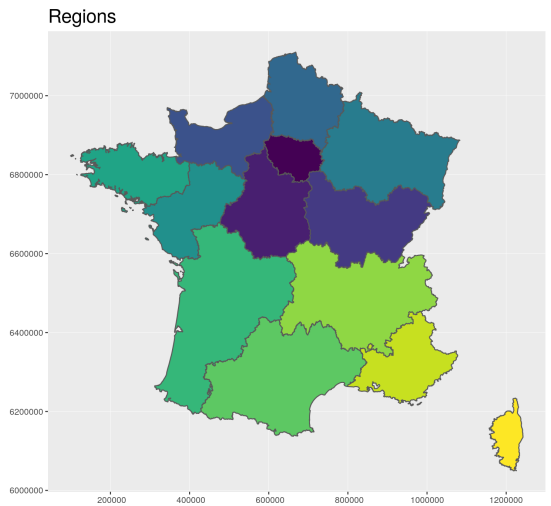


Data manipulation with {dplyr}

- Merging entities with `group_by` + `summarize`

```
region_193 <- departements_193 %>%
  group_by(CODE_REG, NOM_REG) %>%
  summarize()
```

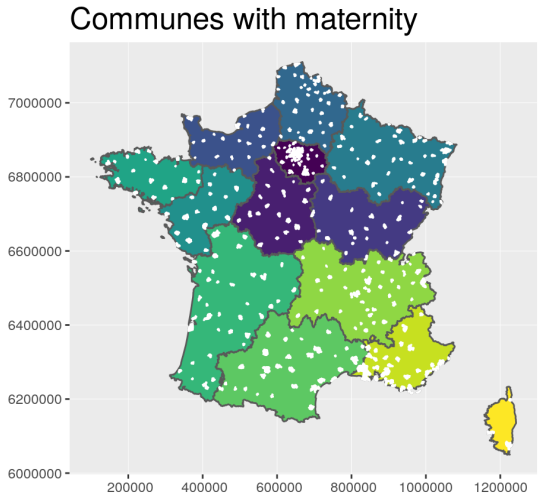




Data manipulation with {dplyr}

- Classic joint of "communes" between CSV file and shapefile with `*_join`
  - Exercise with 2016 data from 'Maternite_2004-2016.csv' and 'COMMUNE.shp'

```
# Join database with shapefile by attributes
maternites_l93 <-
 communes_l93 %>%
 inner_join(data_maternite,
            by = "INSEE_COM") %>%
 st_transform(2154)
```



Data manipulation with {dplyr}

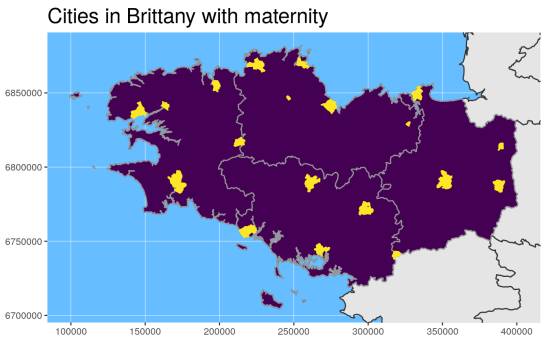
- Classic joint of "communes" between CSV file and shapefile with `*_join`.
  - Exercise with 2016 data from 'Maternite_2004-2016.csv' and 'COMMUNE.shp'.

Spatial join

Realize the spatial intersection of the layers with `st_intersection`

- *A kind of `inner_join` based on geographical positions*
- Points, polygons, lines <=> polygons

```
maternites_Bret_l93 <-
  maternites_l93 %>%
  st_intersection(Bret_l93)
```

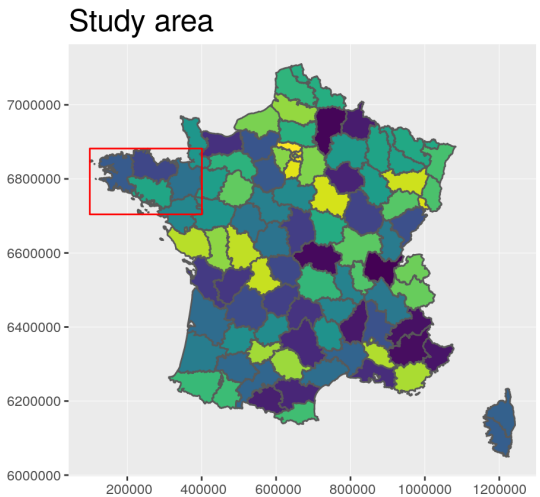


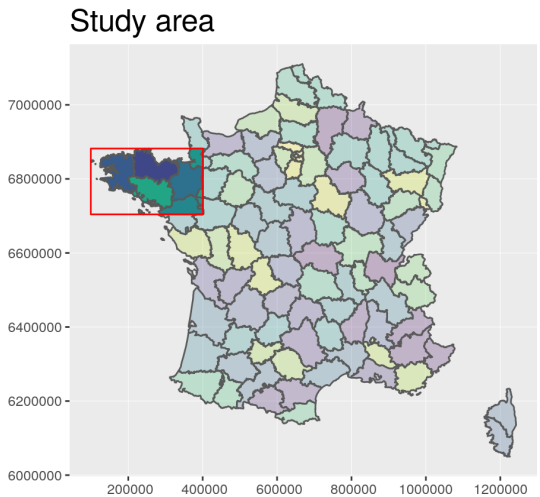

```
#> Maternities

#> [1] "NOM_COM"     "INSEE_COM"
"an"           "Code_Postal" "n"
#> [6] "geometry"

#> Intersection with region

#> [1] "NOM_COM"     "INSEE_COM"
"an"           "Code_Postal" "n"
#> [6] "CODE_DEPT"   "NOM_DEPT"
"NOM_REG"      "geometry"
```

Spatial join

`st_intersection` solves geometry intersections

```
area_l93 <- st_sf(geom = st_sfc(st_polygon(list(matrix(
  c(99217, 6704195,  400572, 6704195,  400572, 6881964,
    99217, 6881964, 99217, 6704195), byrow = TRUE, ncol = 2)))),
  crs = st_crs(departements_l93))

studyarea_l93 <- st_intersection(departements_l93, area_l93)
```

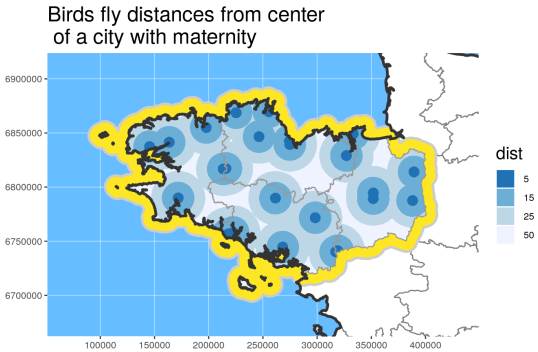




Geomatics Operations

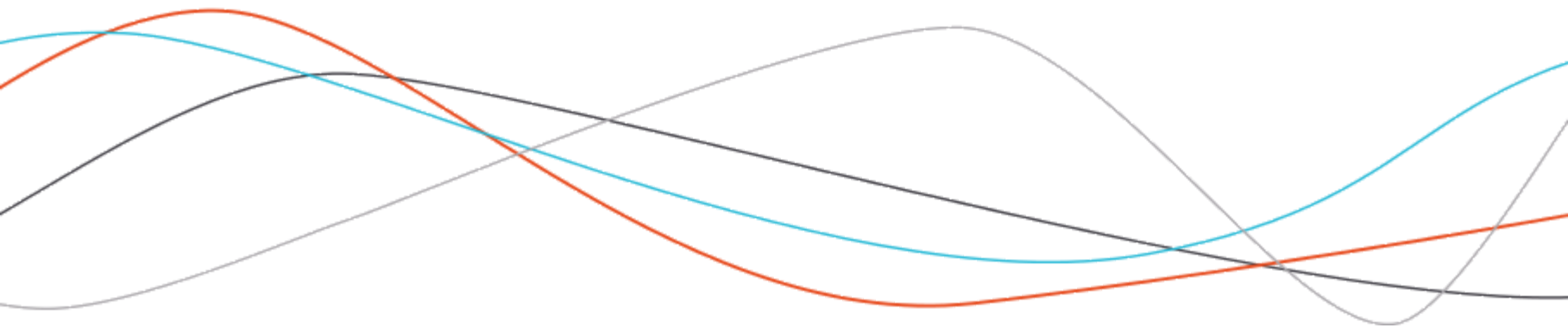
- Get centroids from polygons with `st_centroid`
- Buffer area calculation with `st_buffer`
- Similar to `left_join` for spatial with `st_join`
- Union with `st_union`
- Difference with `st_difference`

```
maternites_centroid_Bret_l93 <-
  maternites_Bret_l93 %>%
  st_centroid()

Bret_buffer10_l93 <-
  Bret_l93 %>%
  st_buffer(
    dist =
      units::set_units(10, km)
  ) %>%
  st_cast() # can be useful
```



Mapping rasters with R

Rasters



Mapping and spatial analysis with R

- Past: {sp} + {rgdal} + {rgeos} + {raster}
- Present: {sf} + {raster}
- Future: {sf} + {stars}

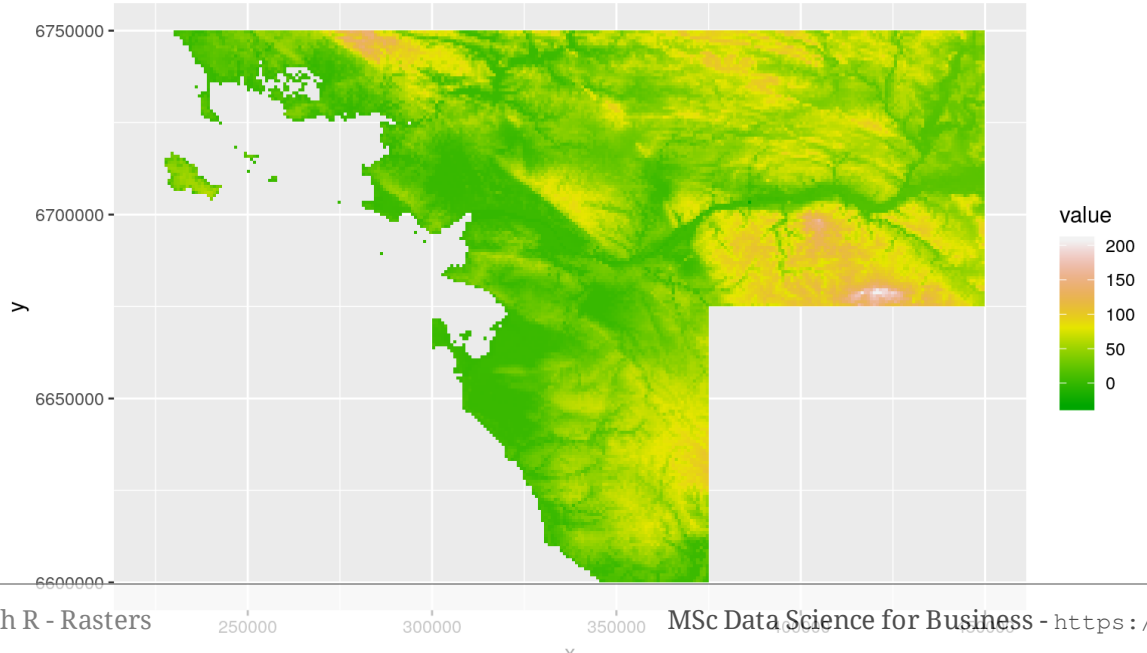
```
# Preparation d'un jeu de données de type {sf}
# dataWD <- "data"
departement_44_l93 <- st_read(
  dsn = file.path(dataWD, "carto-data-orig/departements"),
  layer = "DEPARTEMENT",
  quiet = TRUE) %>%
  st_transform(2154) %>%
  filter(CODE_DEPT == 44)
```

Read raster files

Read using function `raster` :

- All formats handled by GDAL (http://www.gdal.org/)
  - GTiff, HDF5, netCDF, WMS, ENVI, PNG, ...

```
# dataWD <- "data"
Alti_l93 <- raster(file.path(dataWD, "carto-data-
orig/alti/Alti_mosaic_L93.tif"))
```



Preambule: Projections

- Get the coordinate reference system of a raster layer: `projection`

```
projection(Alti_l93)
```

```
#> [1] "+proj=lcc +lat_1=49 +lat_2=44 +lat_0=46.5 +lon_0=3 +x_0=700000
+y_0=6600000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs"
```

- Need to work with the "proj" type string (the coordinate system transformation software: https://proj4.org/about.html)

```
st_crs(departement_44_l93)$proj4string
```

```
#> [1] "+proj=lcc +lat_1=49 +lat_2=44 +lat_0=46.5 +lon_0=3 +x_0=700000
+y_0=6600000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs"
```

```
proj4string(departement_44_sp_l93)
```

```
#> [1] "+proj=lcc +lat_1=49 +lat_2=44 +lat_0=46.5 +lon_0=3 +x_0=700000
+y_0=6600000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs"
```

Preambule: Projections

- Projecting a raster into a new coordinate system: `projectRaster`
  Attention! The projection of a raster causes changes to the original data. Use raw data as much as possible!

```
projection(Alti_l93)
```

```
#> [1] "+proj=lcc +lat_1=49 +lat_2=44 +lat_0=46.5 +lon_0=3 +x_0=700000 +y_0=6600000
+ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs"
```

```
# Convert to geographic coordinates
Alti_wgs84 <- projectRaster(Alti_l93, crs = "+datum=WGS84 +proj=longlat")
projection(Alti_wgs84)
```

```
#> [1] "+datum=WGS84 +proj=longlat +ellps=WGS84 +towgs84=0,0,0"
```

Preambule: Projections

Attention

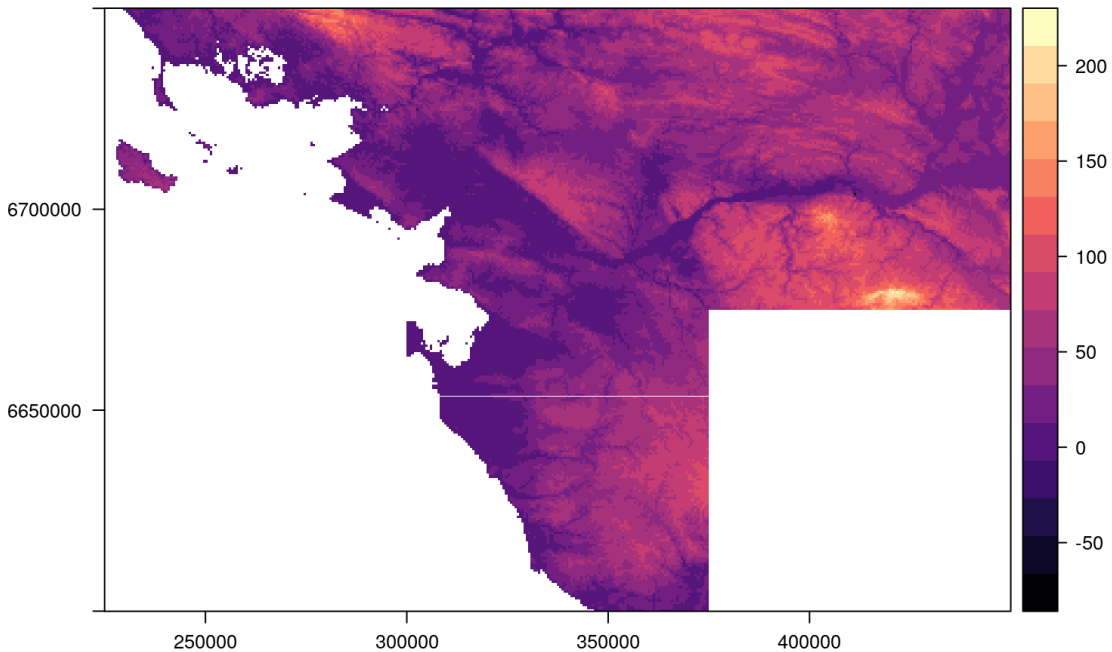
- For some operations, {raster} does not check the coordinate reference system!
  - This can lead to empty extraction results
- For others, it is verified on the basis of the "proj" string
  - Problems for projections with different possible string (*e.g.* lat1/lat2)

```
+proj=lcc +lat_1=49 +lat_2=44 +lat_0=46.5 +lon_0=3 +x_0=700000
+y_0=6600000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs

+proj=lcc +lat_1=44 +lat_2=49 +lat_0=46.5 +lon_0=3 +x_0=700000
+y_0=6600000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
```

Preambule: Draw a raster with {rasterVis}

use `levelplot` ...

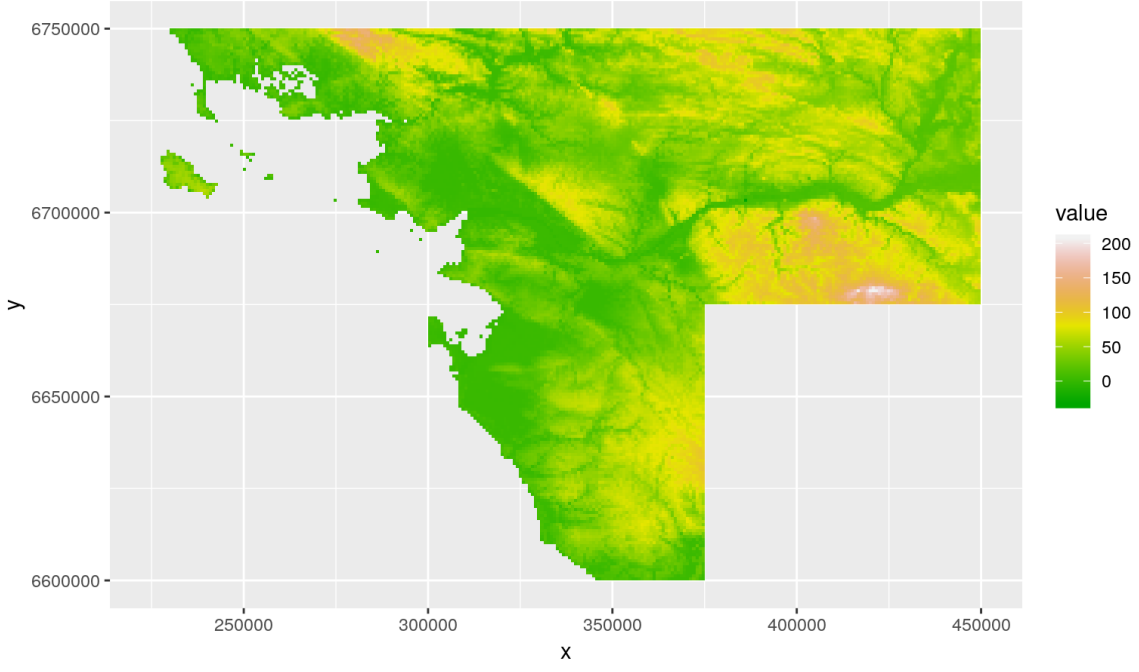
```
levelplot(Alti_l93, margin = FALSE)
```



Preambule: Draw a raster with {rasterVis}

... or `gplot` (with one `g`) to add to {ggplot2}

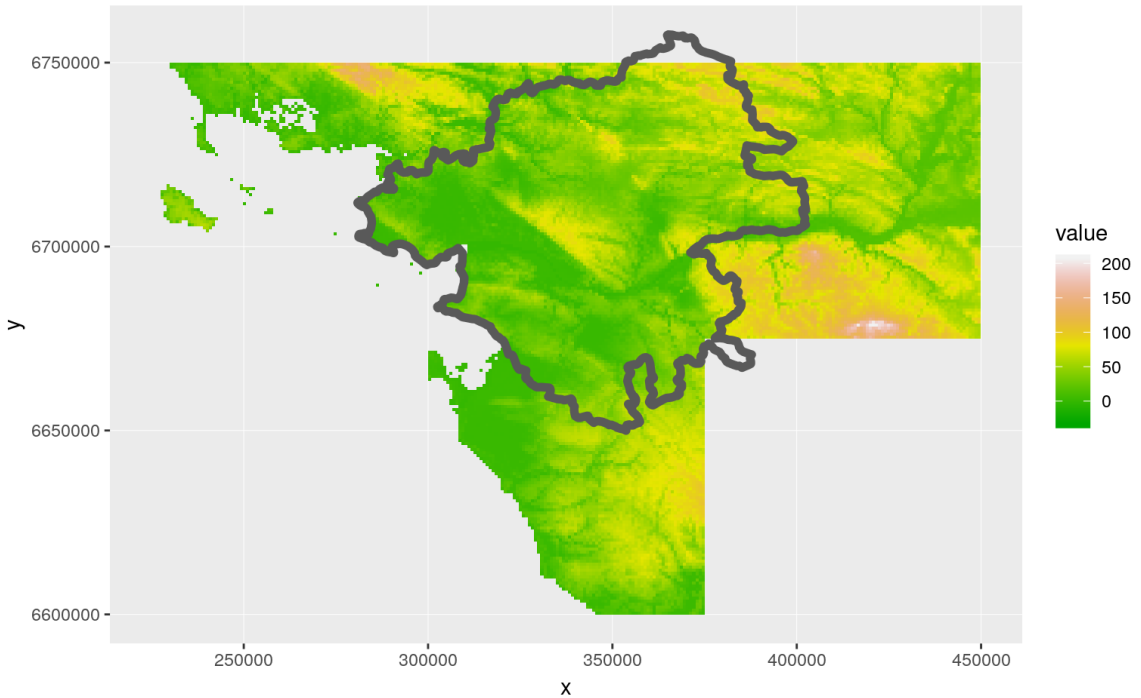
```
gplot(Alti_l93) +
  geom_tile(aes(fill = value)) +
  scale_fill_gradientn(colours = terrain.colors(20), na.value = NA) +
  coord_equal()
```



Crop according to an area or polygon with `crop`

- Be careful, a raster is always rectangular!

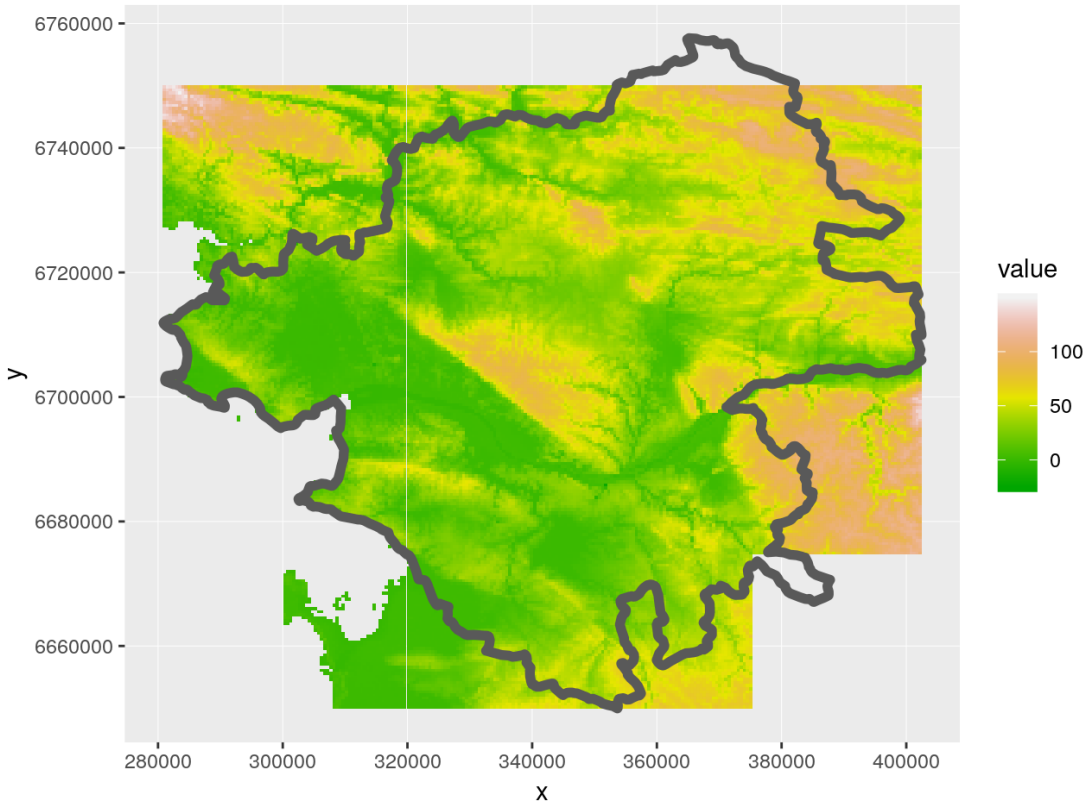
What will happen with `crop` between "Alti_l93" and "departement_44_l93"?



Crop according to an area or polygon with `crop`

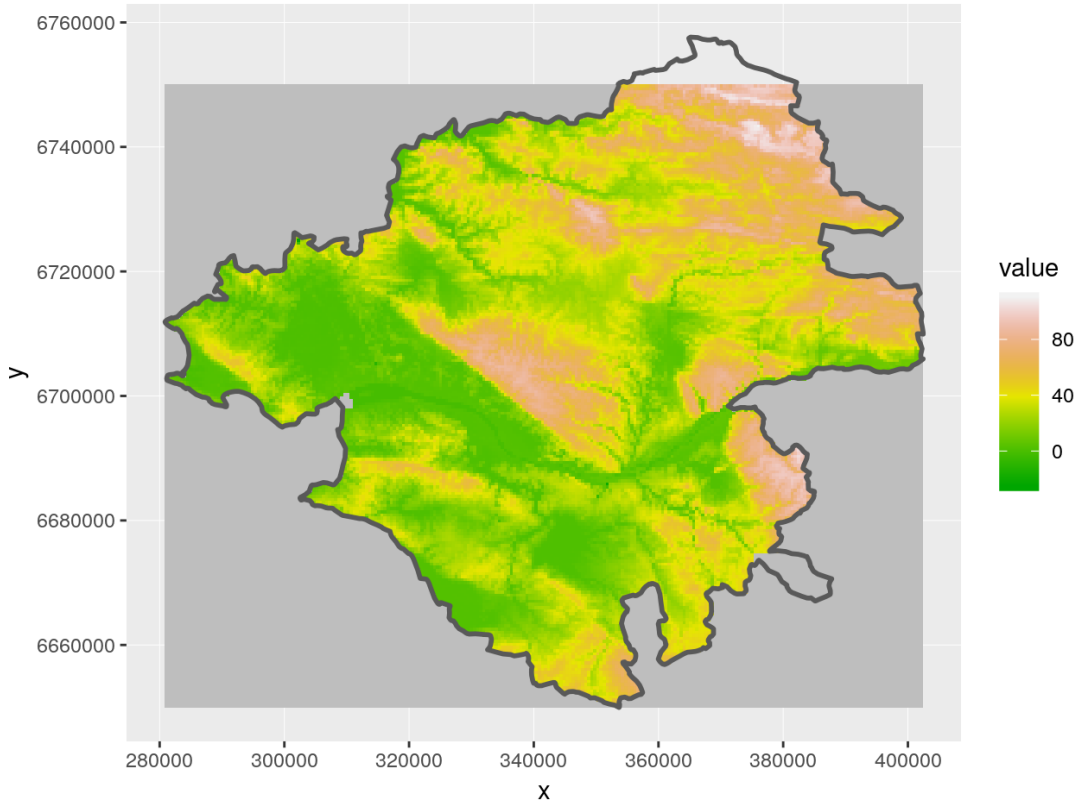
- Be careful, a raster is always rectangular!

```
Alti_44_l93 <- crop(Alti_l93, departement_44_l93) # {sf} ok
```



Mask an area using `mask`

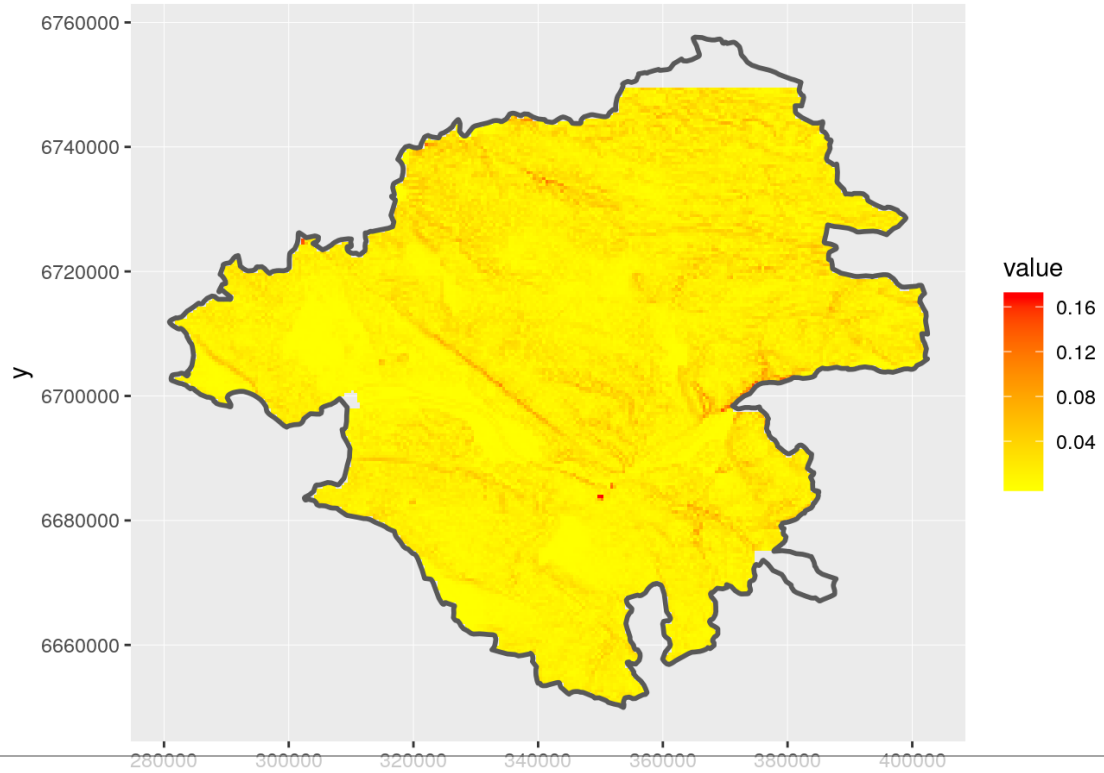
```
Alti_44_mask_l93 <- mask(Alti_44_l93, departement_44_l93) # {sf} ok
```



Calculate topographic parameters using `terrain`

- Slope calculation
  - Comparaison between a pixel and its neighbors (4 or 8)

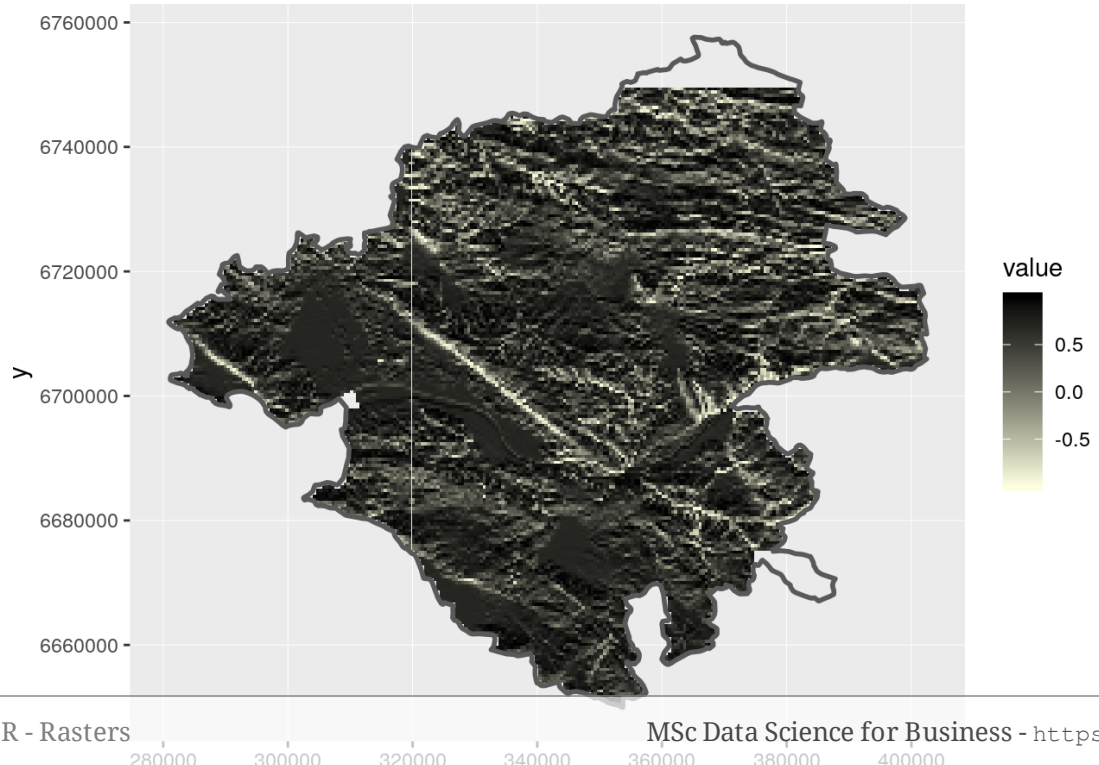
```
slope_44_l93 <- terrain(Alti_44_mask_l93, opt = "slope")
```



Calculate shade using `hillShade`

- Requires `terrain(opt = "aspect") + terrain(opt = "slope")`

```
aspect_44_l93 <- terrain(Alti_44_mask_l93, opt = "aspect")
hillShade_44_wgs84  <- hillShade(
  slope = slope_44_l93 * 40, aspect = aspect_44_l93)
```



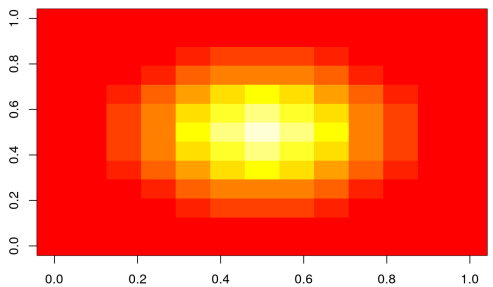
Shade using {rayshader}

- Realistic shade: *https://github.com/tylermorganwall/rayshader*

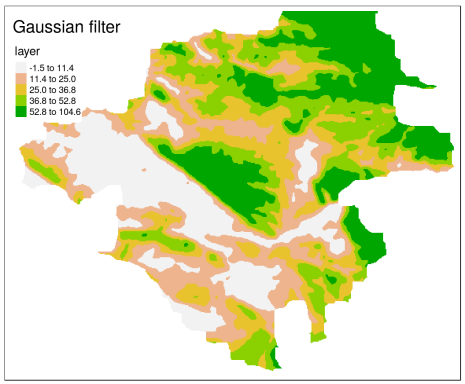
Focal calculations

- Generalization of the `terrain` function
- Calculations on a pixel according to the values of the neighbors
  - Focal window: size and shape
  - Weight of neighboring pixels in the calculation
  - *e.g.* Gaussian filter: round window, Gaussian distribution of weights

```
# focal window
gf <- focalWeight(
  Alti_44_mask_l93, 500, "Gauss")
```




```
Alti_gaussian <-
focal(Alti_44_mask_l93, w = gf)
```

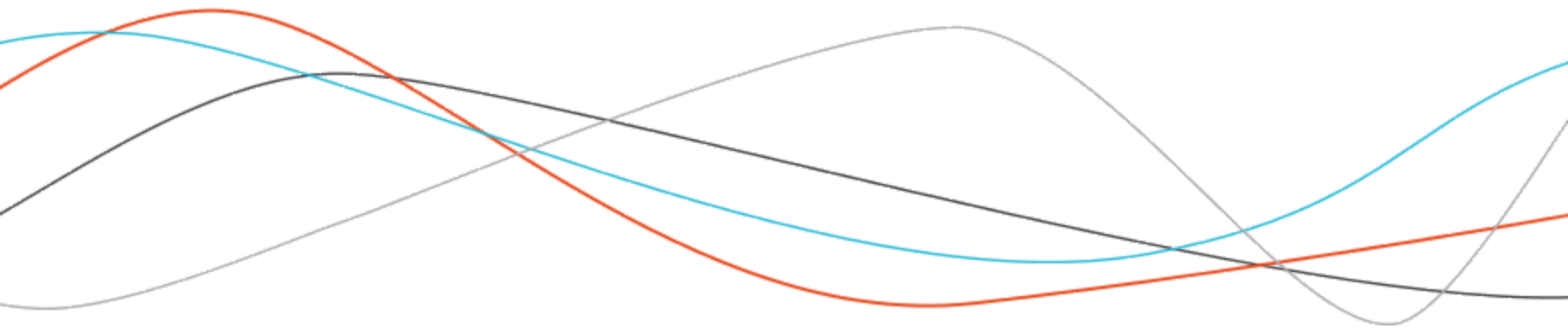


Tip: save on disk

- If a raster is bigger than the RAM available, it is read directly from the disk
  - Allows operations on large rasters
- Most {raster} functions contain the parameter `filename`
  - Allows to save the raster directly on the disk rather than in the RAM (or in a temporary file if the result should take up too much space in the RAM)
  - `overwrite=TRUE` is required to overwrite an existing file

Map visualisation

Create maps with R



Visualise geographical data

Packages for visualisation:

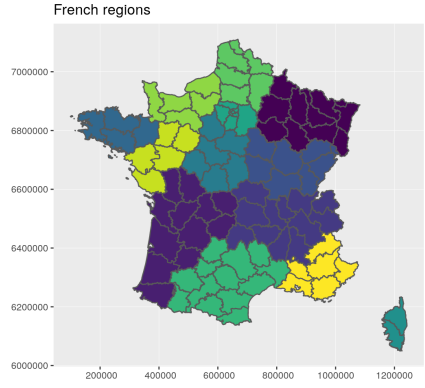
- static: {ggplot2}, {ggspatial}, {tmap}, {cartography}, {rgl}
- interactive: {mapview}, {leaflet}

```
# dataWD <- "data"
# vector
departements_l93 <- st_read(
  dsn = file.path(dataWD, "carto-data-orig/departements"),
  layer = "DEPARTEMENT",
  quiet = TRUE) %>%
  st_transform(2154)
# raster
Alti_l93 <- raster(file.path(dataWD, "carto-data-
orig/alti/Alti_mosaic_L93.tif"))
```

Maps with {ggplot2}

- Version 3.0+
- `geom_sf` : recognize geometry type
- `coord_sf` : axes limits and *crs*. Always define parameters.

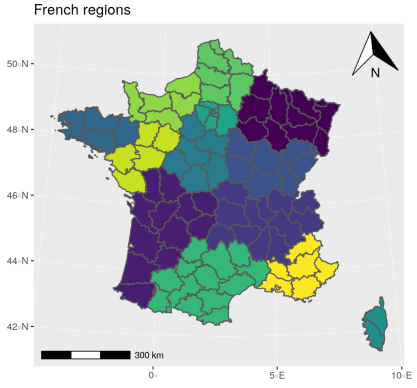
```
ggplot(departements_193) +
  geom_sf(aes(fill = NOM_REG)) +
  coord_sf(
    crs =
st_crs(departements_193),
    datum =
st_crs(departements_193)) +
  scale_fill_viridis_d() +
  guides(fill = FALSE) +
  ggtitle("French regions")
```



Maps with {ggspatial}

- Combines with `geom_sf`
- `annotation_north_arrow` to add north arrow
- `annotation_scale` to add scale

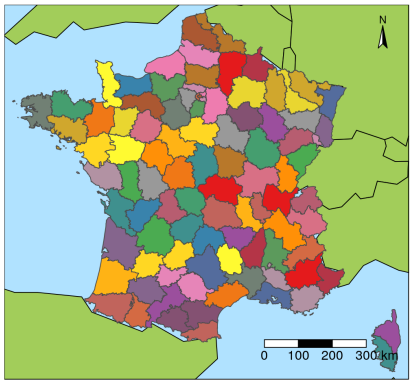
```
ggplot(departements_193) +
  geom_sf(aes(fill = NOM_REG)) +
  coord_sf(
    crs =
st_crs(departements_193),
    datum = 4326) +
  scale_fill_viridis_d() +

annotation_north_arrow(which_north
 = "true") +
  annotation_scale() +
  guides(fill = FALSE) +
  ggtitle("French regions")
```



Maps with {tmap}

- `tm_shape` for each new layer
  - `tm_dots` : points
  - `tm_fill` : polygons
  - `tm_text` : text
- `tm_scale_bar`
- `tm_compass`

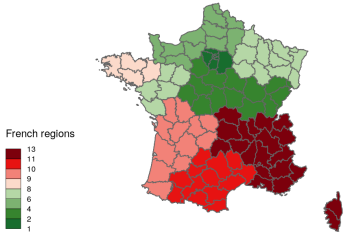
```
tm_shape(Europe) +
  tm_polygons() +
tm_shape(departements_l93,
         is.master = TRUE) +
  tm_fill(col = "NOM_DEPT",
          legend.show = FALSE,
          palette = "Set1") +
  tm_borders("grey30") +
tm_scale_bar(position =
c("right",
"bottom"), size = 1) +
tm_compass(position = c("right",
  "top"), size = 1.8) +
```



Maps with {cartography}

- Add layers to base plot with specific functions using `add=TRUE`
  - `choroLayer` for choropleth maps
  - `propSymbolsLayer` for proportional symbols
  - `gradLinkLayer` for flow maps

```
cols <- carto.pal(
  pal1 = "green.pal", n1 = 4,
  pal2 = "red.pal", n2 = 4)

plot(st_geometry(departements_l93))

choroLayer(x = departements_l93,
  var = "NUM_REG",
  col = cols,
  border = "grey40",
  legend.title.txt = "French
regions",
  add = TRUE)
```

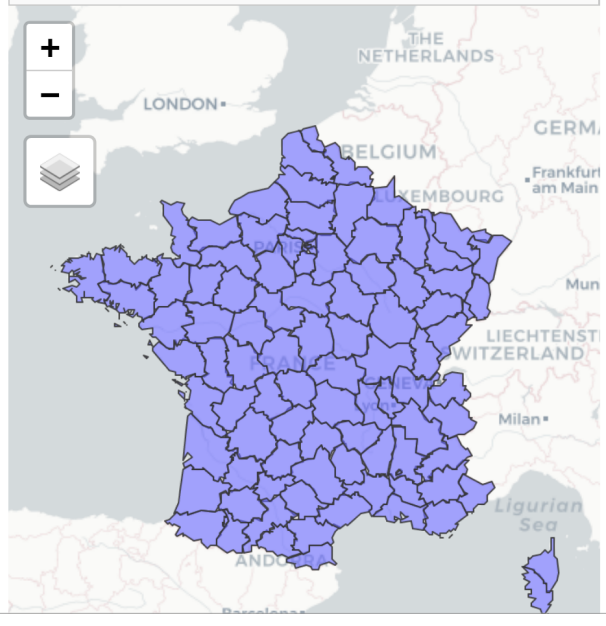


Explore data interactively

- Use {mapview} to quickly visualise data interactively
  - During your development process

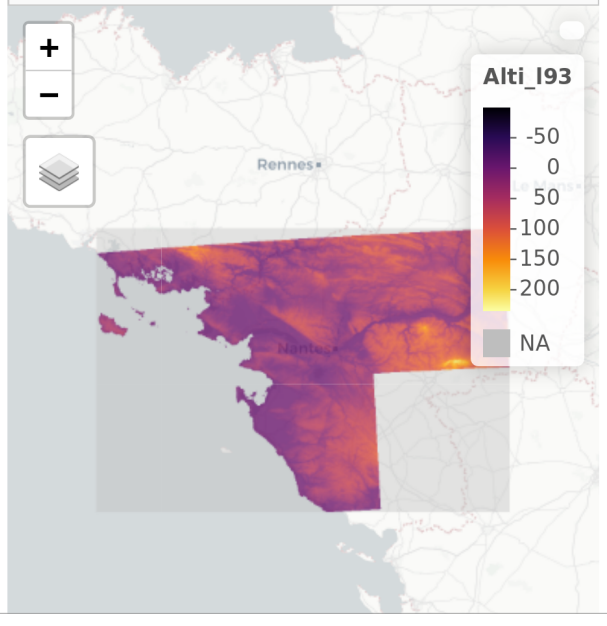
```
# mapview(departements_l93)
```

```
mapview(departements_l93)
```



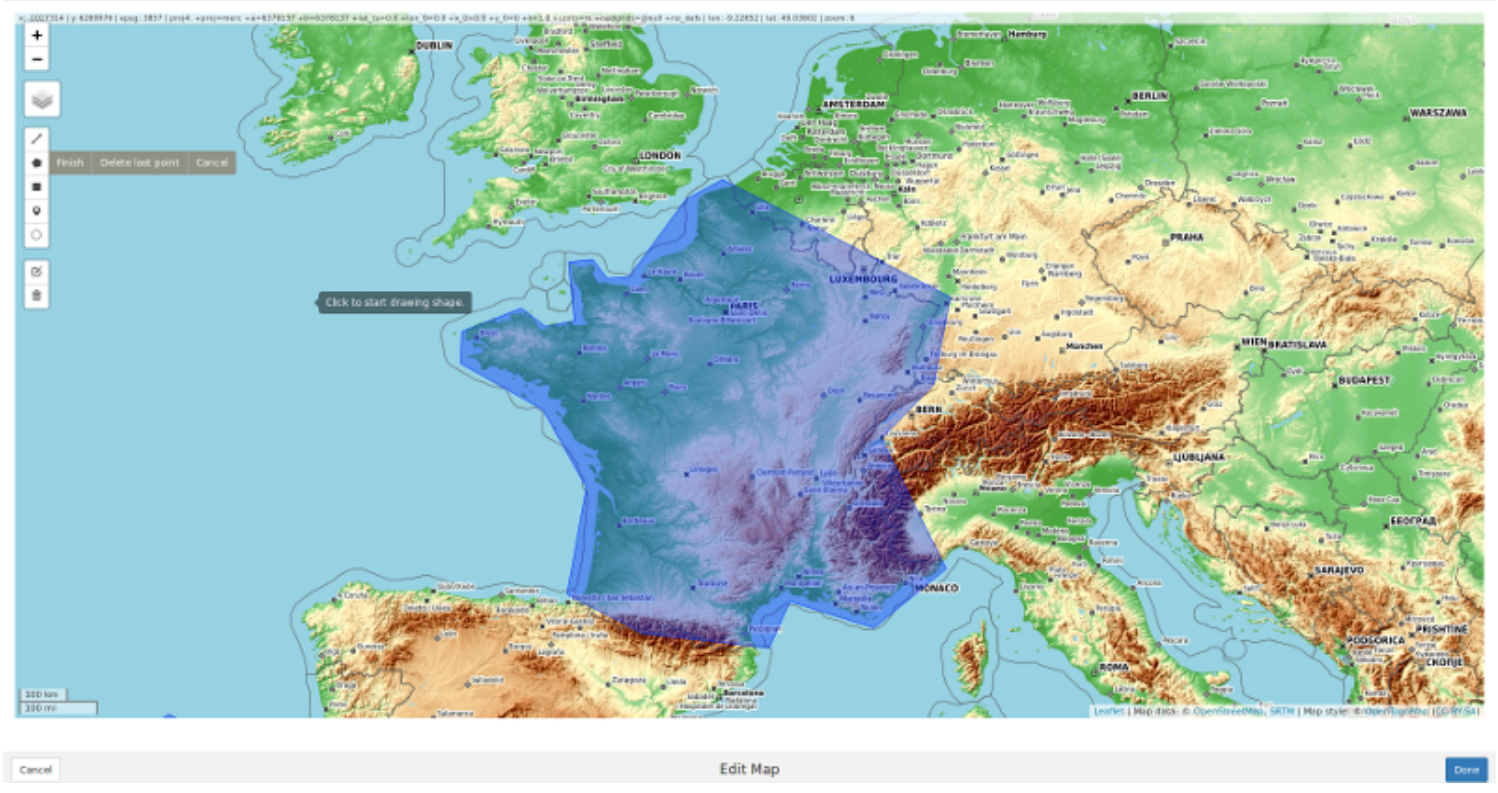

```
# mapview(Alti_l93)
```

```
mapview(Alti_l93)
```



Modify data interactively

- Use {mapedit} in combination with {mapview} to create new spatial dataset interactively



Interactive maps using {leaflet} in HTML

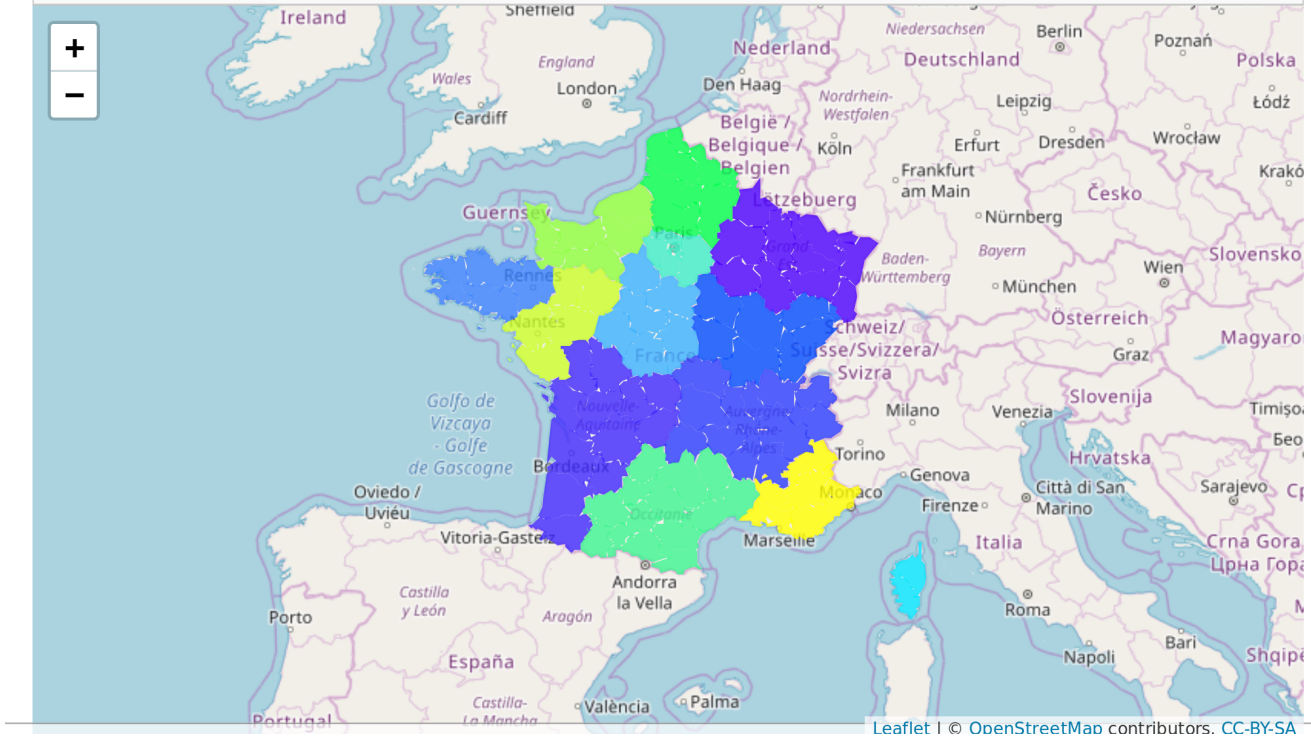
- `addTiles()` : Background maps
- `addMarkers` : Points
- `addPolygons` : Polygons
- Requires `crs = 4326`
- *Create palettes for colours before leaflet*

```
departements_wgs84 <-
  st_transform(departements_l93, crs = 4326)

factpal <- colorFactor(topo.colors(5), departements_wgs84$NOM_REG)

m <- leaflet() %>%
  addTiles() %>%
  addPolygons(data = departements_wgs84,
              color = ~factpal(NOM_REG),
              fillOpacity = 0.8, stroke = FALSE)
```

Interactive maps using {leaflet} in HTML

```
m
```

```
m
```

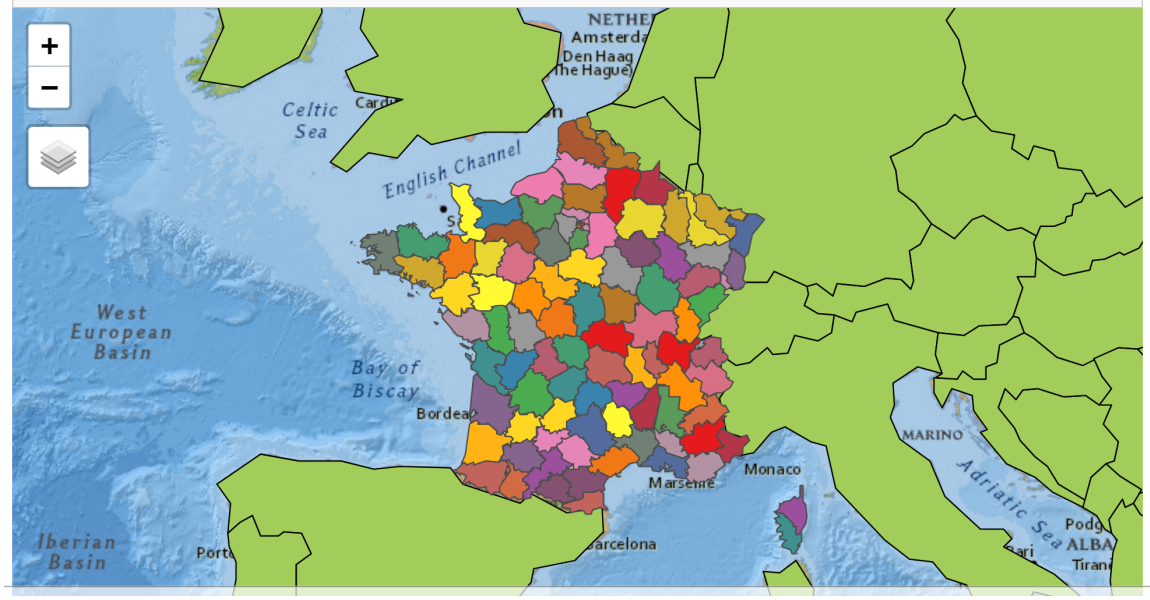


Interactive maps using {leaflet} and {tmap}

- Magick function in {tmap}

```
tmap_leaflet(tm)
```

```
tmap_leaflet(tm)
```



Sébastien Rochette

sebastien@thinkr.fr

06 74 01 65 14