

Étude préventive des avalanches : modélisation et applications

...

ARNAUD Thomas 32447

SOMMAIRE

- 1 - Introduction, présentation de l' avalanche, ses caractéristiques. Statistiques.
- 2 - Exploration du manteau neigeux
- 3 - Développement d'un algorithme de détection des zones à risque en fonction d'un bulletin météo et d'une analyse de terrain.
- 4 - Bilan et limites

1 - Introduction, présentation de l'avalanche, caractéristiques, statistiques.



Photo 3 : Une plaque glissante peut se transformer en avalanche lorsque la force d'entrainement sous l'effet de la gravité devient plus importante que les forces de retenue. (Davos Frauenkirch, décembre 2011; photo SLF/S. Margreth)

1 - Introduction, présentation de l'avalanche, caractéristiques, statistiques.



Avalanche "aérosol" (SkiPass)



Avalanche "plaqué" (European Avalanche Warning Services)



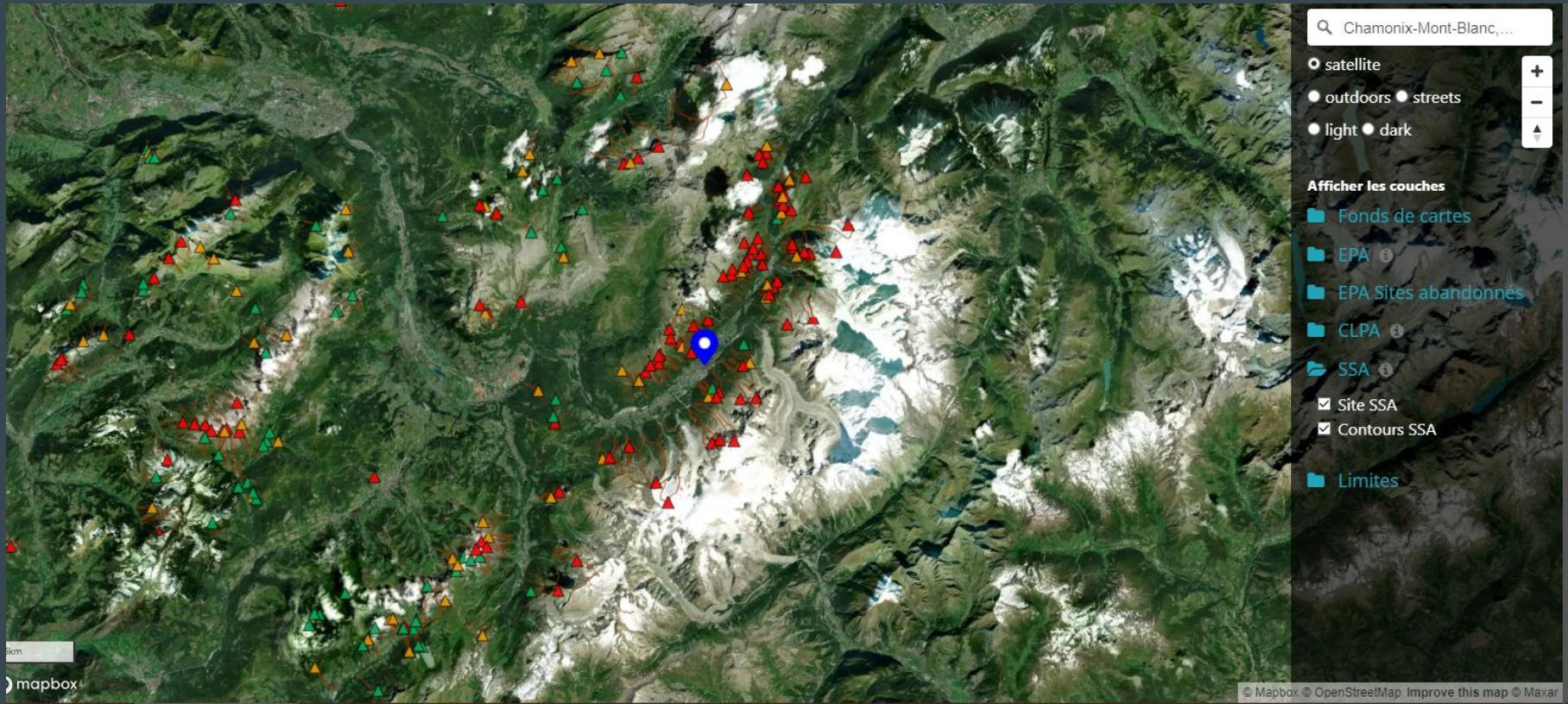
Avalanche "Printemps" (Zapiks)

1 - Introduction, présentation de l'avalanche, caractéristiques, statistiques.



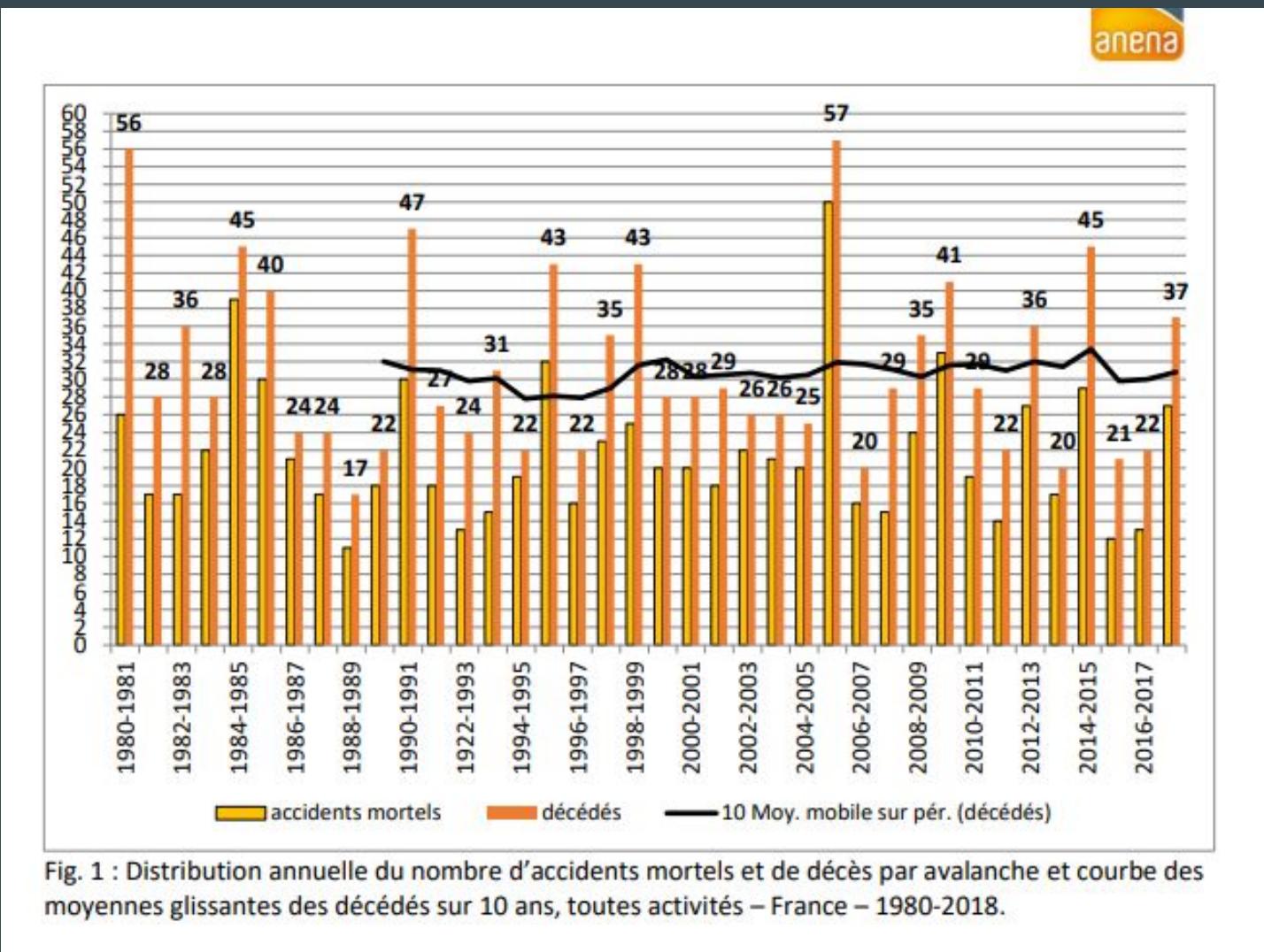
Carte SSA (sites sensibles aux avalanches) de la France. (map.avalanches.fr)

1 - Introduction, présentation de l'avalanche, caractéristiques, statistiques.



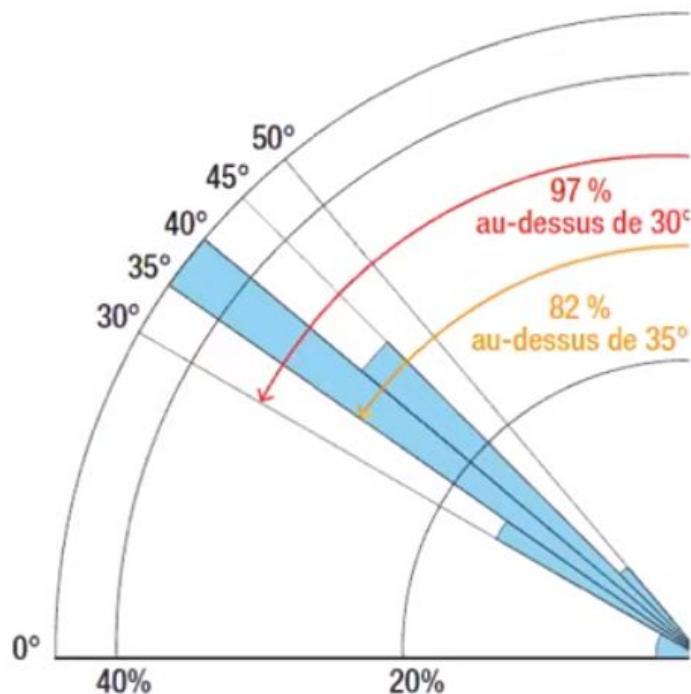
Carte SSA autour de la station de Chamonix.

1 - Introduction, présentation de l'avalanche, caractéristiques, statistiques.



1 - Introduction, présentation de l'avalanche, caractéristiques, statistiques.

INCLINAISON DES PENTES ET ACCIDENTS

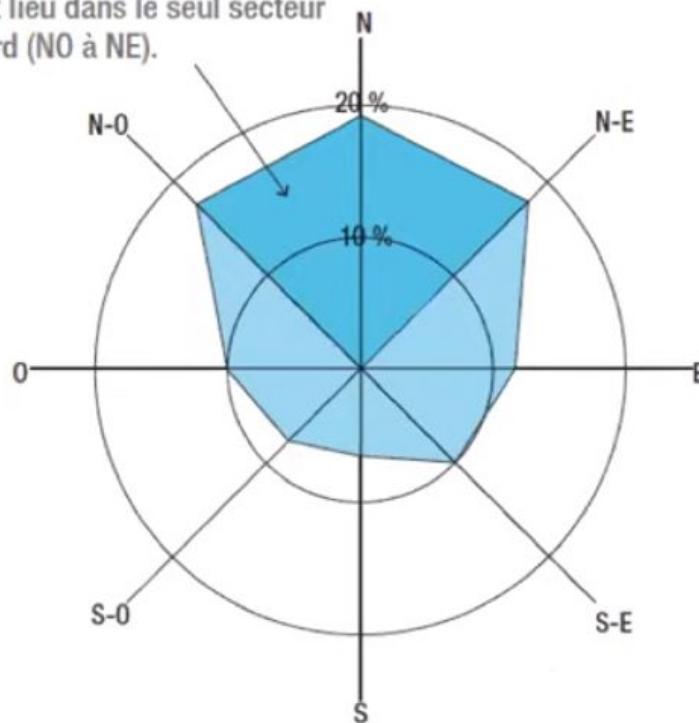


43 % des avalanches ayant emporté des personnes se sont déclenchées dans des pentes comprises entre 35° et 40°.

Statistiques de 1997 à 2017. Source : K. Winkler, B. Zweifel, C. Marty et Franck Techel, « Schnee und Lawinen in den Schweizer Alpen, Hydrologisches Jahr 2017/2018 », SLF-WSL, Davos, 2018.

ACCIDENTS ET ORIENTATION

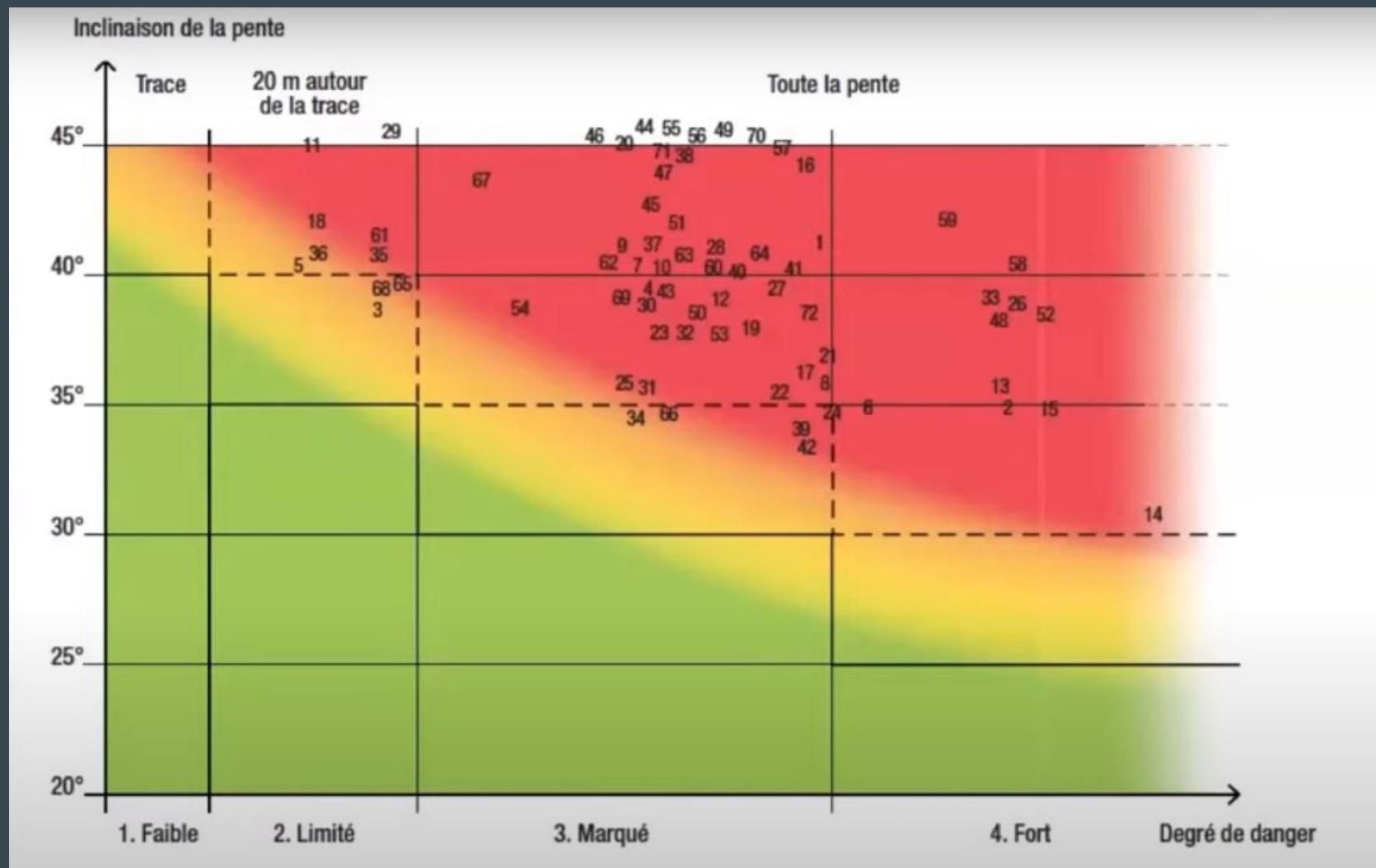
Près de 60 % des accidents ont lieu dans le seul secteur nord (NO à NE).



En bleu figure, selon l'orientation, le pourcentage (entre 1997 et 2017) des accidents impliquant des personnes en Suisse, en fonction de l'orientation des pentes.

Source : Frank Techel et Gian Darms, *Schnee und Lawinen in den Schweizer Alpen, Hydrologisches Jahr 2017/2018*, SLF, Davos, 2018.

1 - Introduction, présentation de l'avalanche, caractéristiques, statistiques.



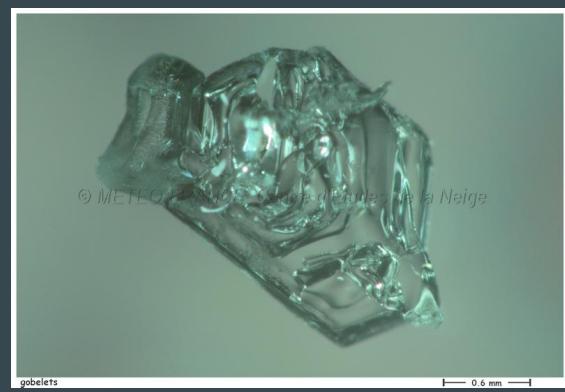
2 - Exploration du manteau neigeux



Neige fraîche (CNRM)



Grains fins (CNRM)

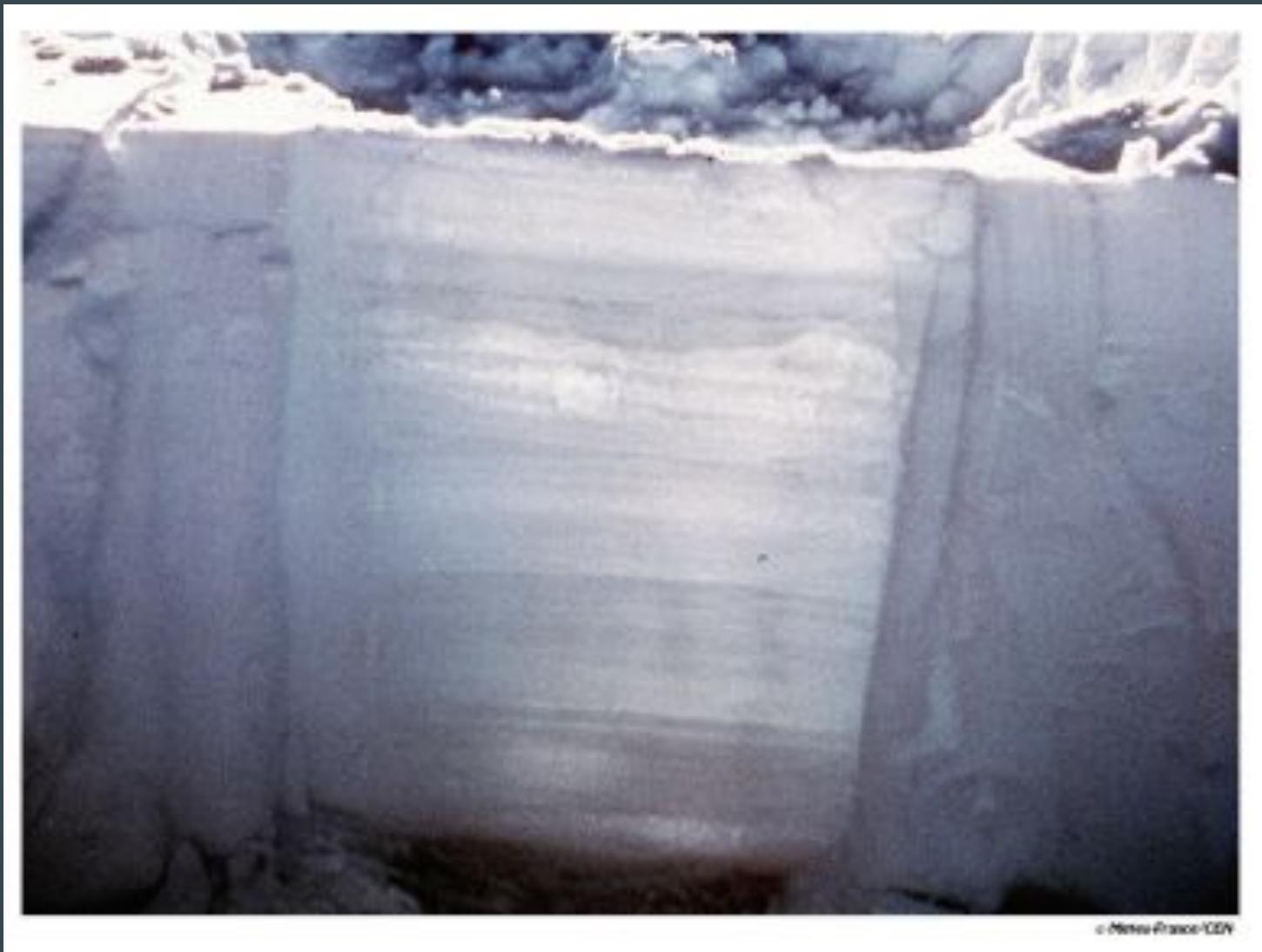


Gobelets (CNRM)

2 - Exploration du manteau neigeux

	Neige récente	Neige récente	Neige récente	Neige évoluée	Neige évoluée	Neige évoluée	Neige évoluée	Neige évoluée	
Type	Neige Fraîche	Neige Roulée	Givre de surface	Grains fins	Grains à face plane	Gobelets	Grains ronds Humides	Grains ronds regelés	Neige de culture
Cohésion	-	- - -	- - -	+	-	- - -	-	+++	
Caractéristiques	Surface, présents après chutes de neige sans vent faible températures	Surface/profondeur, d = 1mm, rarement en couche épaisse, s'accumulent avec le vent.	Croit de la surface en captant les vapeurs d'eau contenues dans l'air. Créer une couche fragile	d = 0.3 mm Sur piste damés ou non, principal composant du manteau neigeux.	d = 0.5mm présent dans les zones peu exposées au soleil.	d = 1 -> 5 mm Présents sur les versants à l'ombre. Persistent en profondeur, créent une couche fragile.	d = 1 mm Pentes sud ensoleillées	L'eau règle formant une plaque très solide	= Grains ronds à plus petit diamètres

2 - Exploration du manteau neigeux



2 - Exploration du manteau neigeux

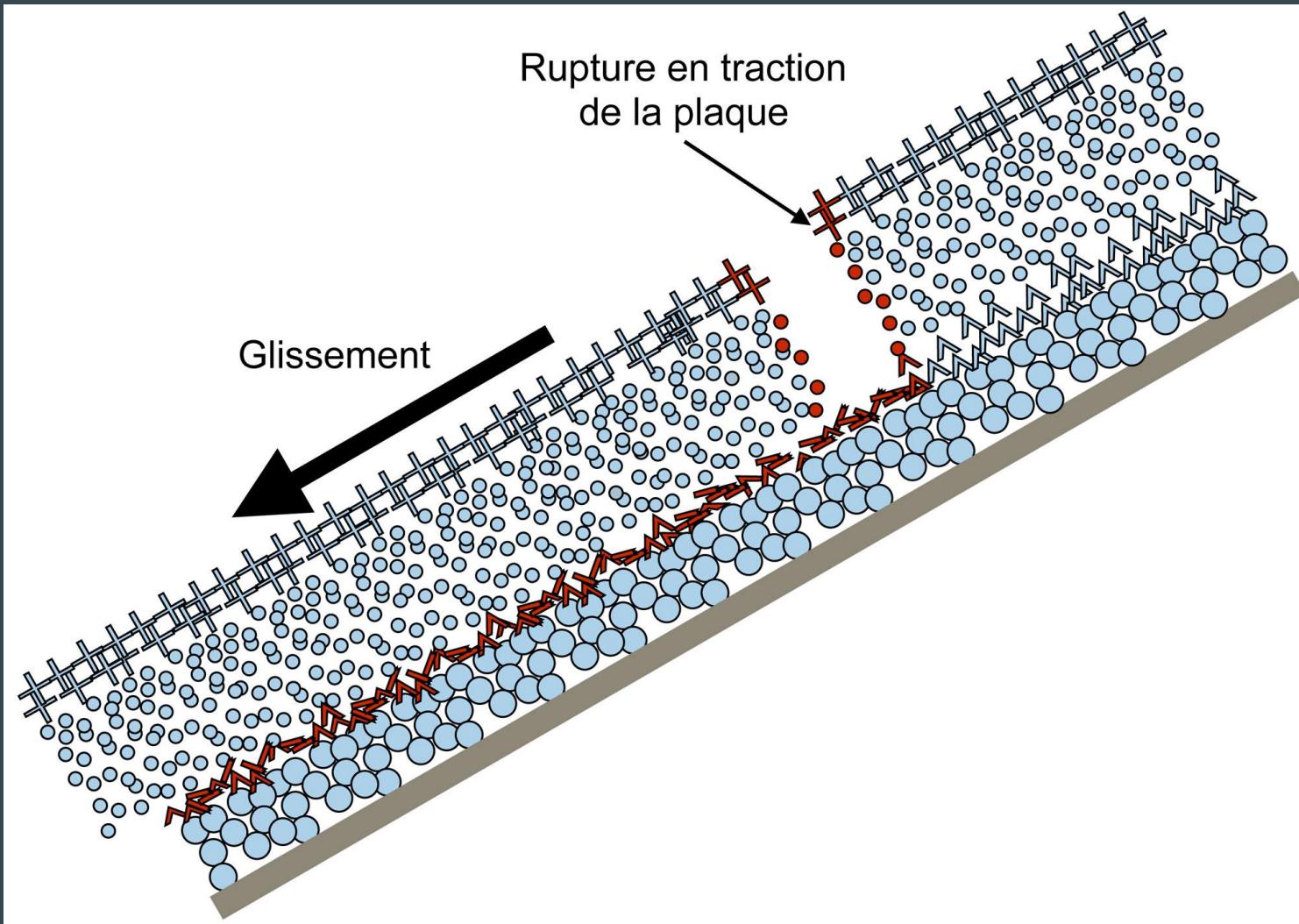
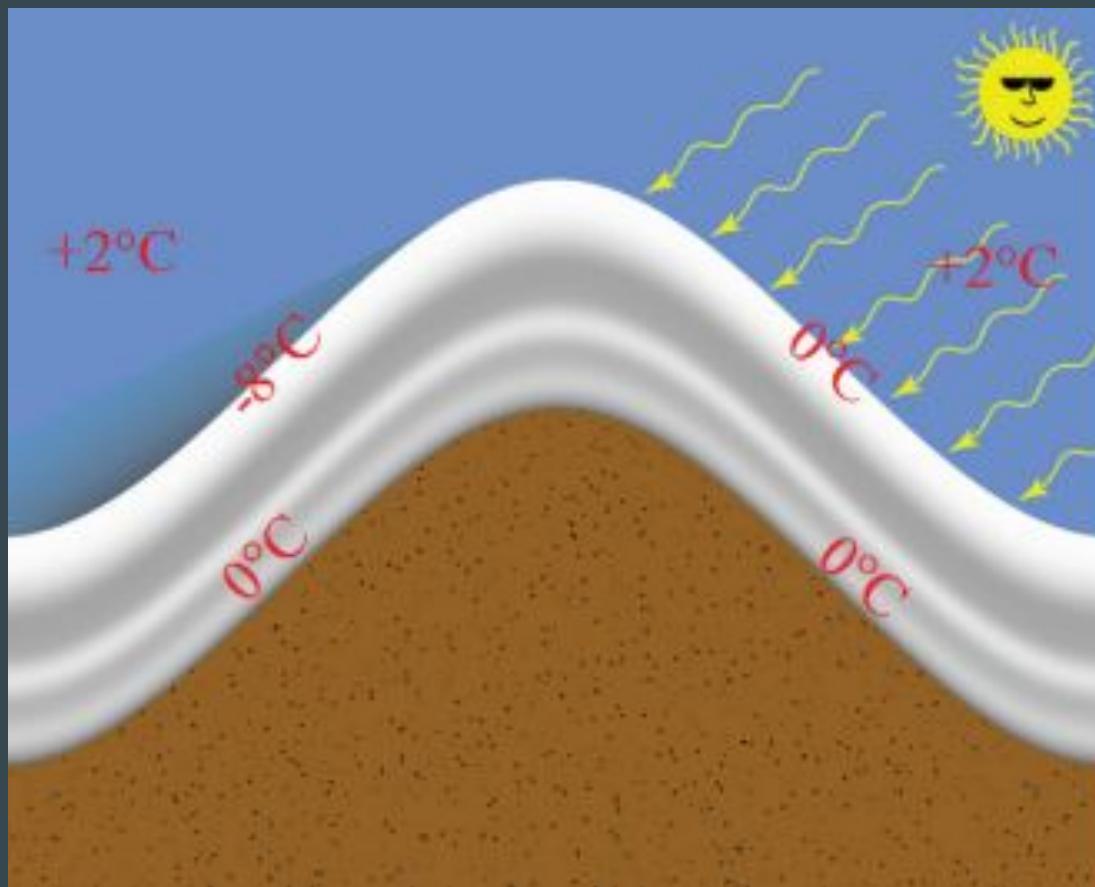
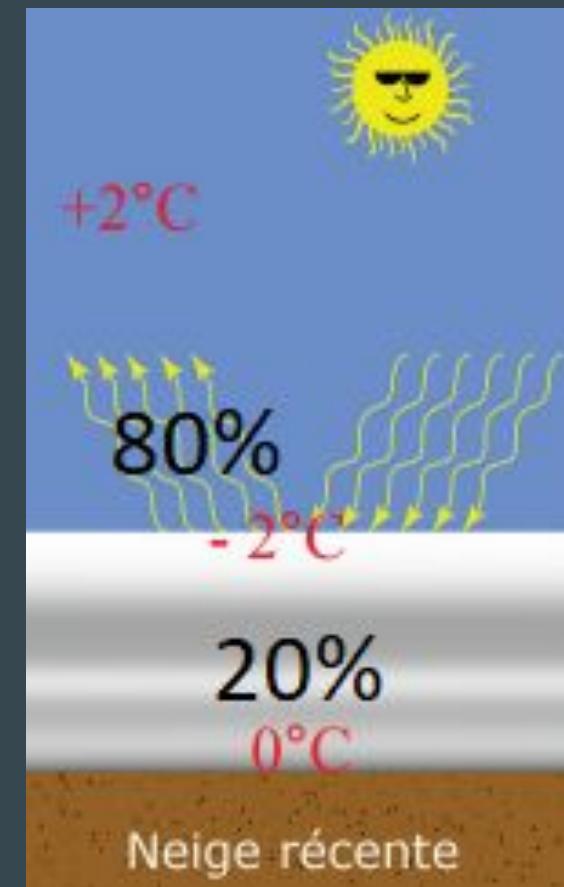


Schéma de rupture (SKI PASS)

2 - Exploration du manteau neigeux



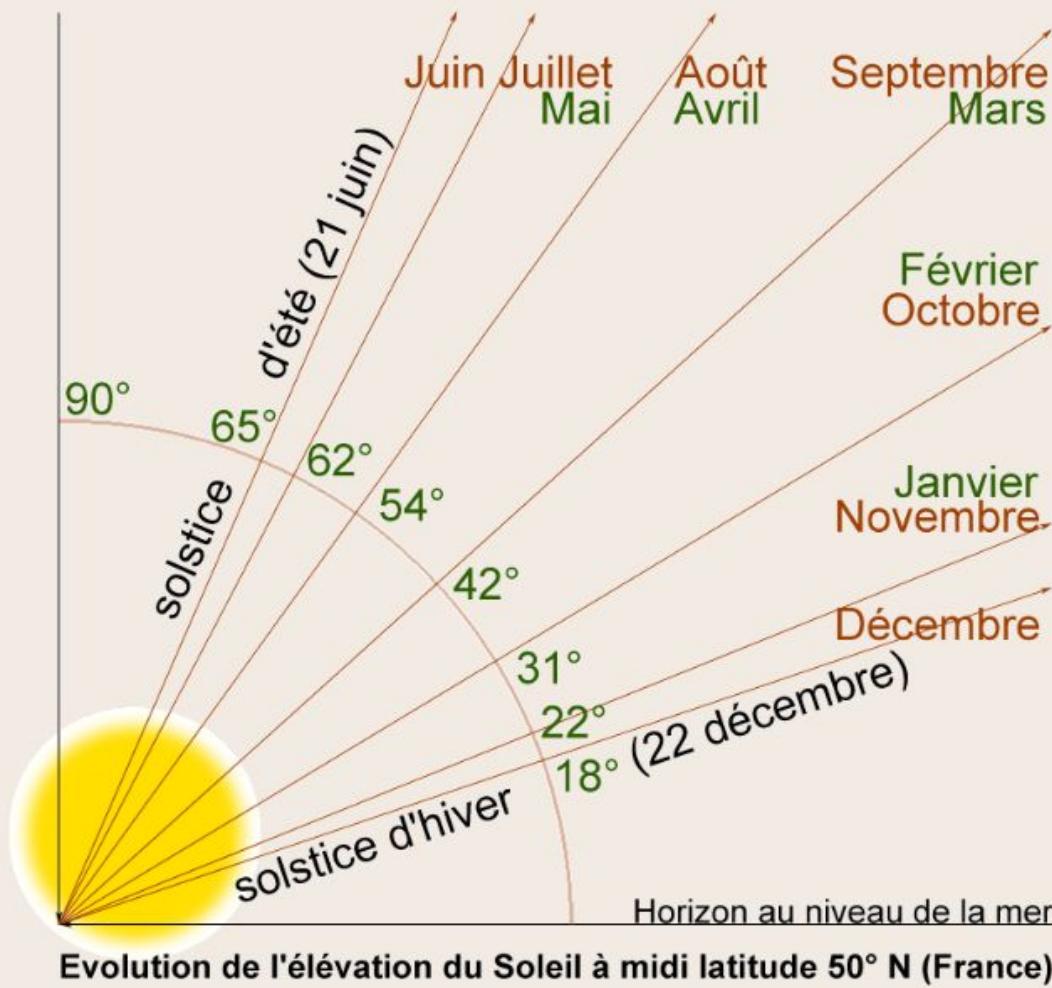
skirando.com



skirando.com

2 - Exploration du manteau neigeux

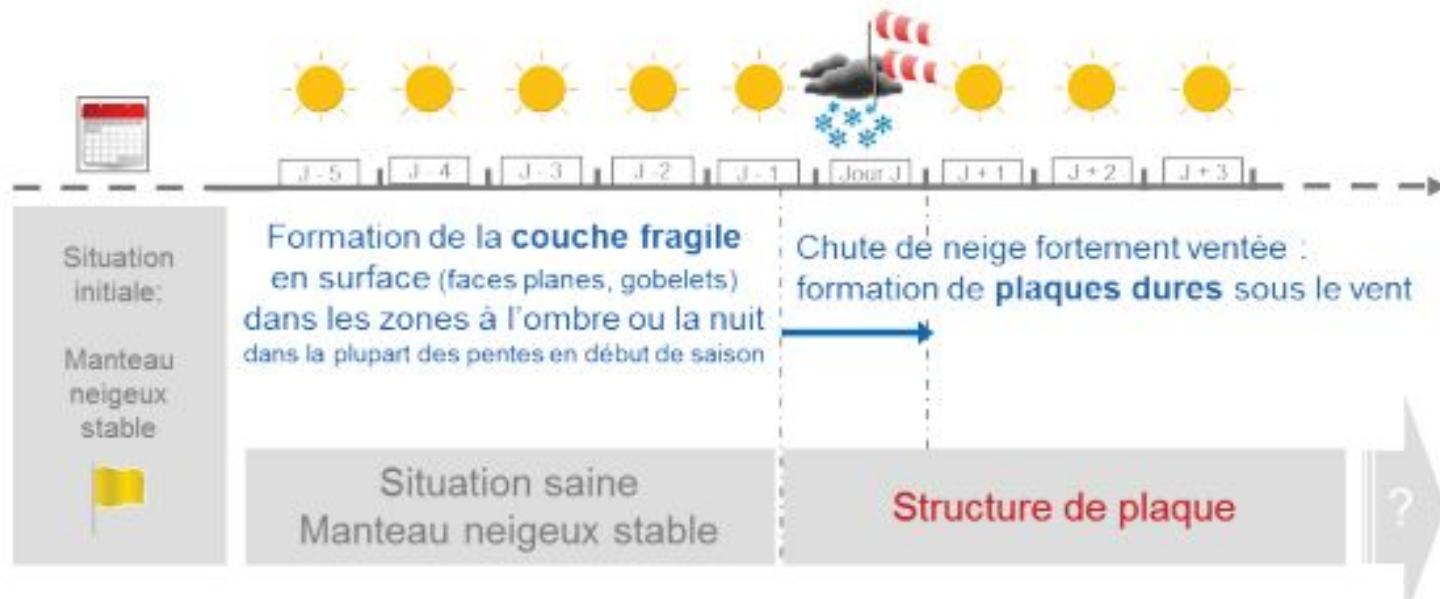
La carte ci-dessous indique l'inclinaison du soleil à midi pour chaque mois de l'année, angle minimum au solstice d'hiver (-+21 déc.), angle maximum au solstice d'été (+-21 juin). Cet direktion est calculée pour la France (latitude 50° Nord). L'angle doit être augmenté de 5° par 800 km lorsque l'on descend au Sud jusqu'à l'équateur (Soleil au plus haut) et soustrait de 5° par 800 km si l'on voyage vers le Nord.



2 - Exploration du manteau neigeux

Exemple n°1. Beau temps froid et sec, puis chute de neige fortement ventée.

Origine de la structure de plaque : Couche fragile : métamorphoses / Plaques : vent.



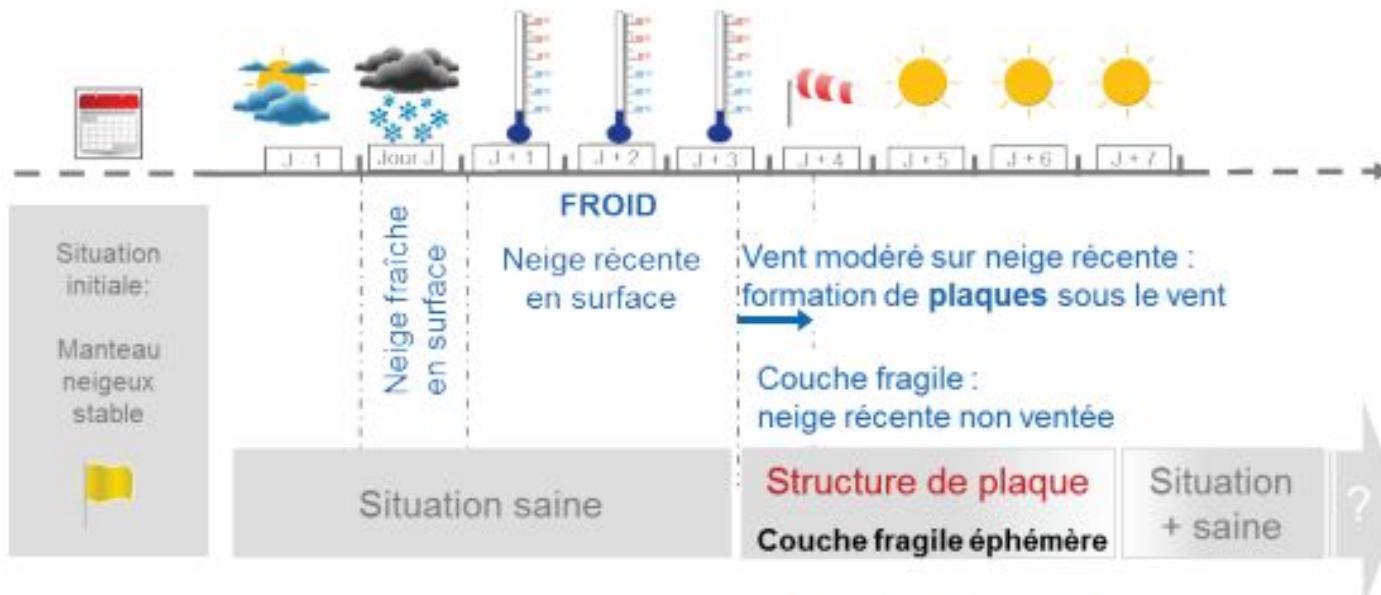
Danger d'avalanche :

- **localisé** (effet du vent)
- **durable** (plusieurs jours)

2 - Exploration du manteau neigeux

Exemple n°2. Chute de neige, puis temps froid et sec, puis vent modéré

Origine de la structure de plaque : Couche fragile : précipitations / Plaques : vent.



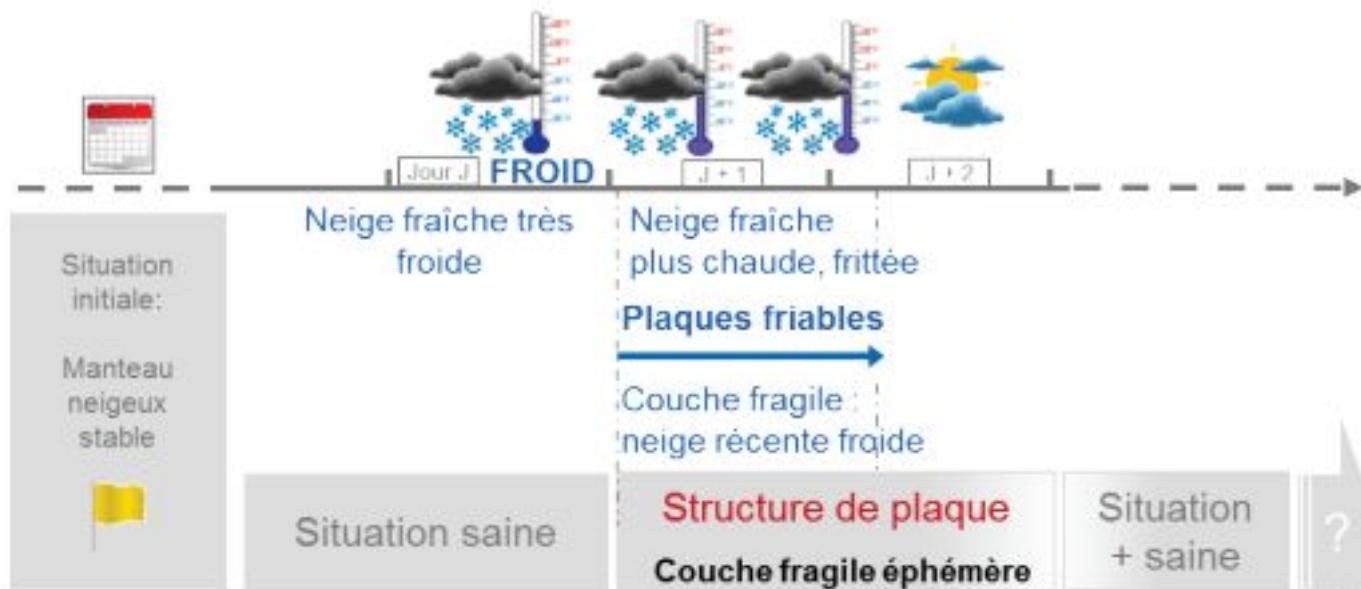
Danger d'avalanche :

- **localisé** (effet du vent)
- **temporaire** (quelques heures à 1 journée)

2 - Exploration du manteau neigeux

Exemple n°3. Chute de neige très peu ventée, froide au départ puis se réchauffant.

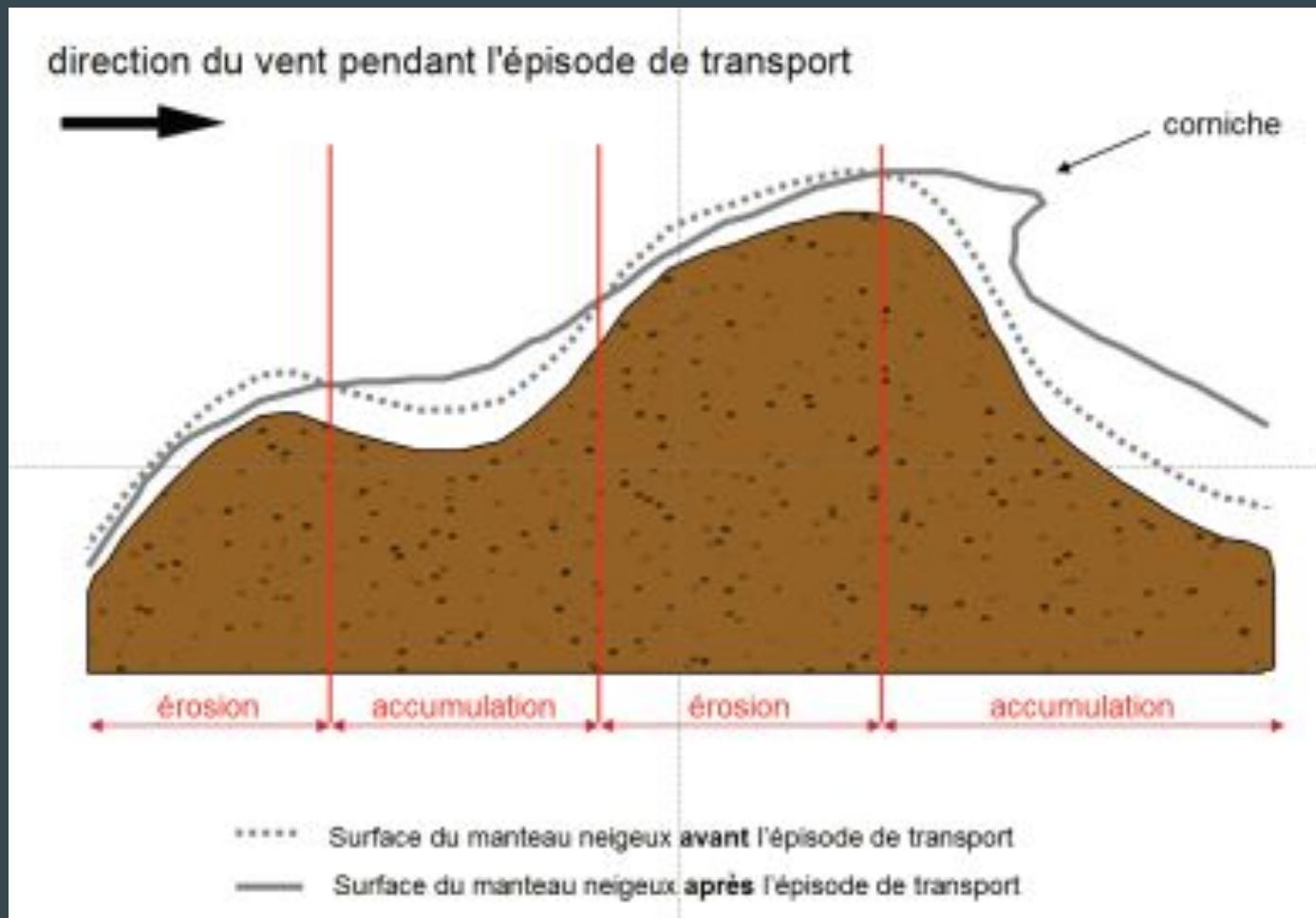
Origine de la structure de plaque : Couche fragile : précipitations / Plaques : précipitations et métamorphoses.



Danger d'avalanche :

- **généralisé** (tranche d'altitude)
- **temporaire** (quelques heures à 1 journée)

2 - Exploration du manteau neigeux



3 - Développement d'un algorithme de détection des zones à risque en fonction d'un bulletin météo et d'une analyse de terrain.

1 -> Algorithme générant un bulletin météo

2 -> Algorithme déterminant le facteur risque avalanche en fonction du bulletin météo.

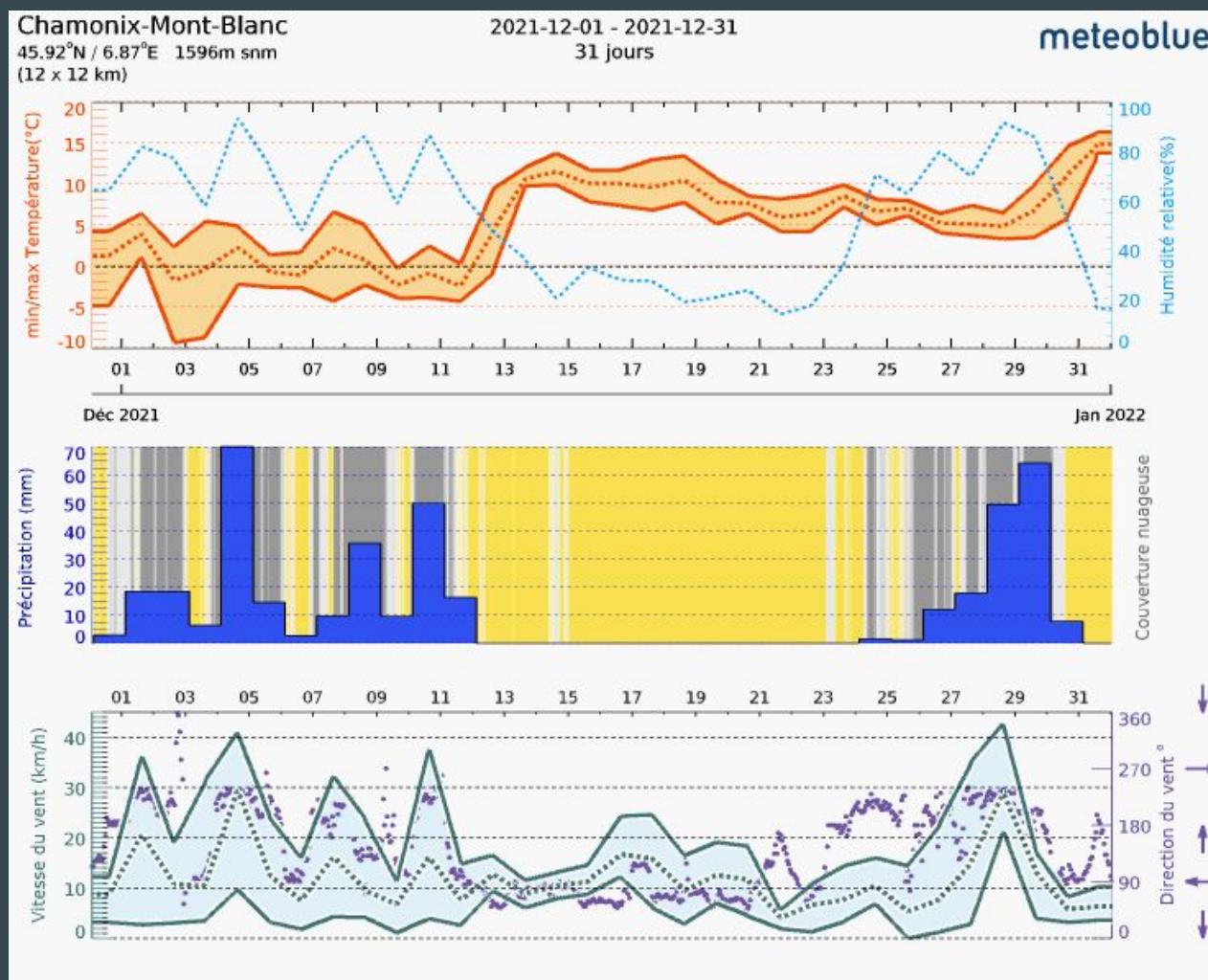
3 -> Algorithme de comparaison de mes estimations avec celles de MétéoFrance.

4 -> Algorithme générant une matrice d'altitude d'une montagne à partir d'un MNT.

5 -> Algorithme de détermination des pentes d'une matrice d'altitude.

6 -> Algorithme de détermination des pentes à risques d'une montagne en fonction du facteur risque avalanche et du bulletin météo.

3.1 - Algorithme de bulletin météo



Décembre 2021

archives meteoblue.com

3.1 - Algorithme de bulletin météo

```
In [22]: GenereBulletinMeteo()
```

```
Out[22]:
```

```
[['Pluie', -9, 37, 3, 173],  
 ['Soleil', 2, 0, 0, 169],  
 ['Soleil', -1, 0, 0, 162],  
 ['Soleil', 0, 0, 3, 156],  
 ['Pluie', -10, 41, 0, 165],  
 ['Soleil', 1, 0, 0, 161]]
```

```
In [23]:
```

3.2 - Algorithme déterminant le facteur risque avalanche

Indice de risque	1	2	3	4
Enneigement	[0;50]	[50;100]	[100;150]	[150;+oo[
Vent	[0;30]	[30;40]	[40;50]	[50;+oo[
IndiceMeteo				

Patterns à risque 4 :

Soleil - Soleil - Neige - Soleil - Soleil [Longtemps]

Neige - Froid - Froid - Vent - Soleil [Jour]

Froid - Neige - Pluie - Pluie [Jour]

Patterns à risques 3 :

Soleil - Soleil - Neige - Soleil [L]

Soleil - Neige - Soleil - Soleil [L]

Neige - Froid - Vent - Soleil [J]

Neige- Vent - Soleil [J]

Neige - Froid - Froid - Vent [J]

Patterns à risque 2 :

3.2 - Algorithme déterminant le facteur risque avalanche

```
In [23]: IndiceRisqueAvalanche()
Out[23]:
(3.0,
 [[{'Pluie': -3, '40': 2, '100': 100}, {'Pluie': -3, '32': 2, '113': 113}, {'Soleil': 12, '0': 0, '108': 108}, {'Soleil': 13, '0': 1, '103': 103}, {'Pluie': -3, '31': 0, '110': 110}, {'Pluie': -6, '49': 0, '121': 121}])
```

3.2 - Algorithme déterminant le facteur risque avalanche

PourcentageIndice :

```
In [31]: PourcentageIndices(100000)
```

```
Out[31]: [15.653, 68.34, 15.968, 0.039]
```

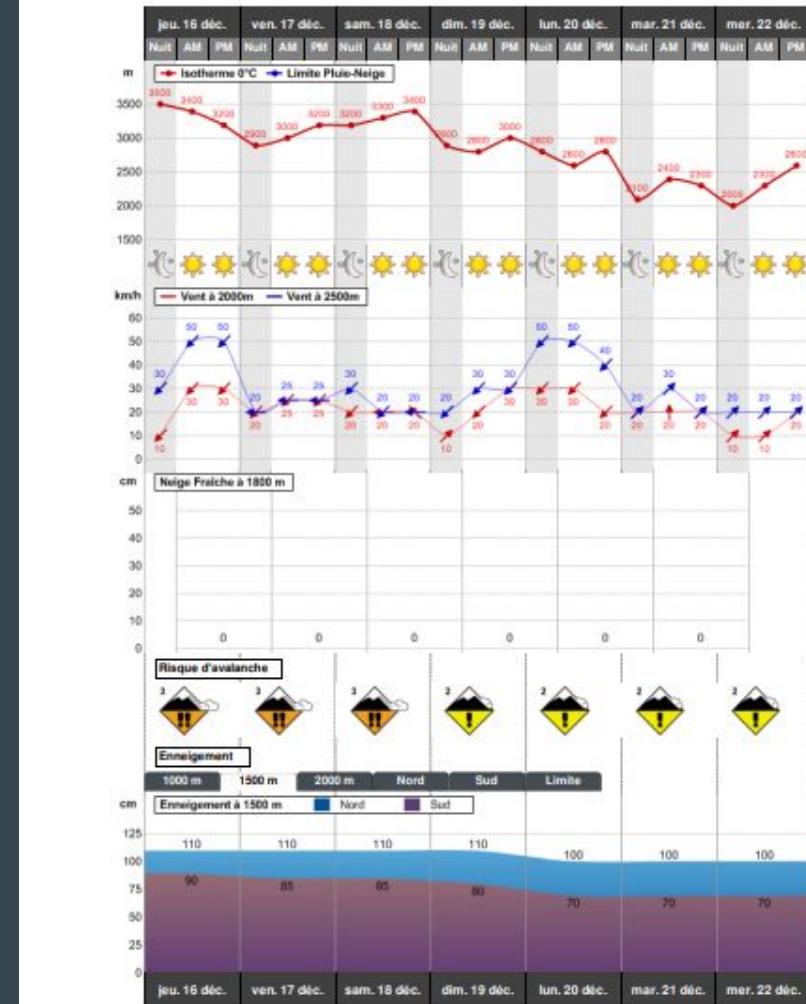
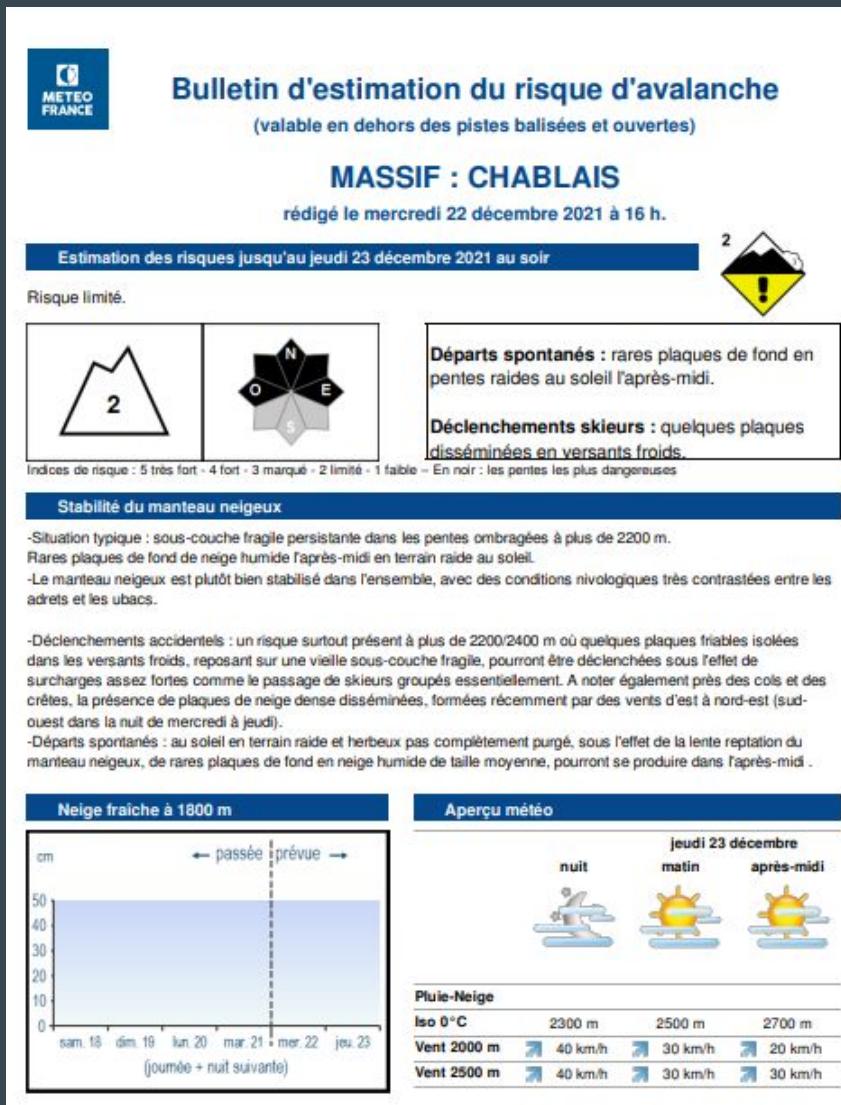
```
In [32]: PourcentageIndices(100000)
```

```
Out[32]: [15.878, 67.972, 16.101, 0.049]
```

```
In [33]: PourcentageIndices(1000000)
```

```
Out[33]: [15.8914, 68.1195, 15.9498, 0.0393]
```

4.3 - Algorithme de comparaison de mes estimations avec celles de MétéoFrance.

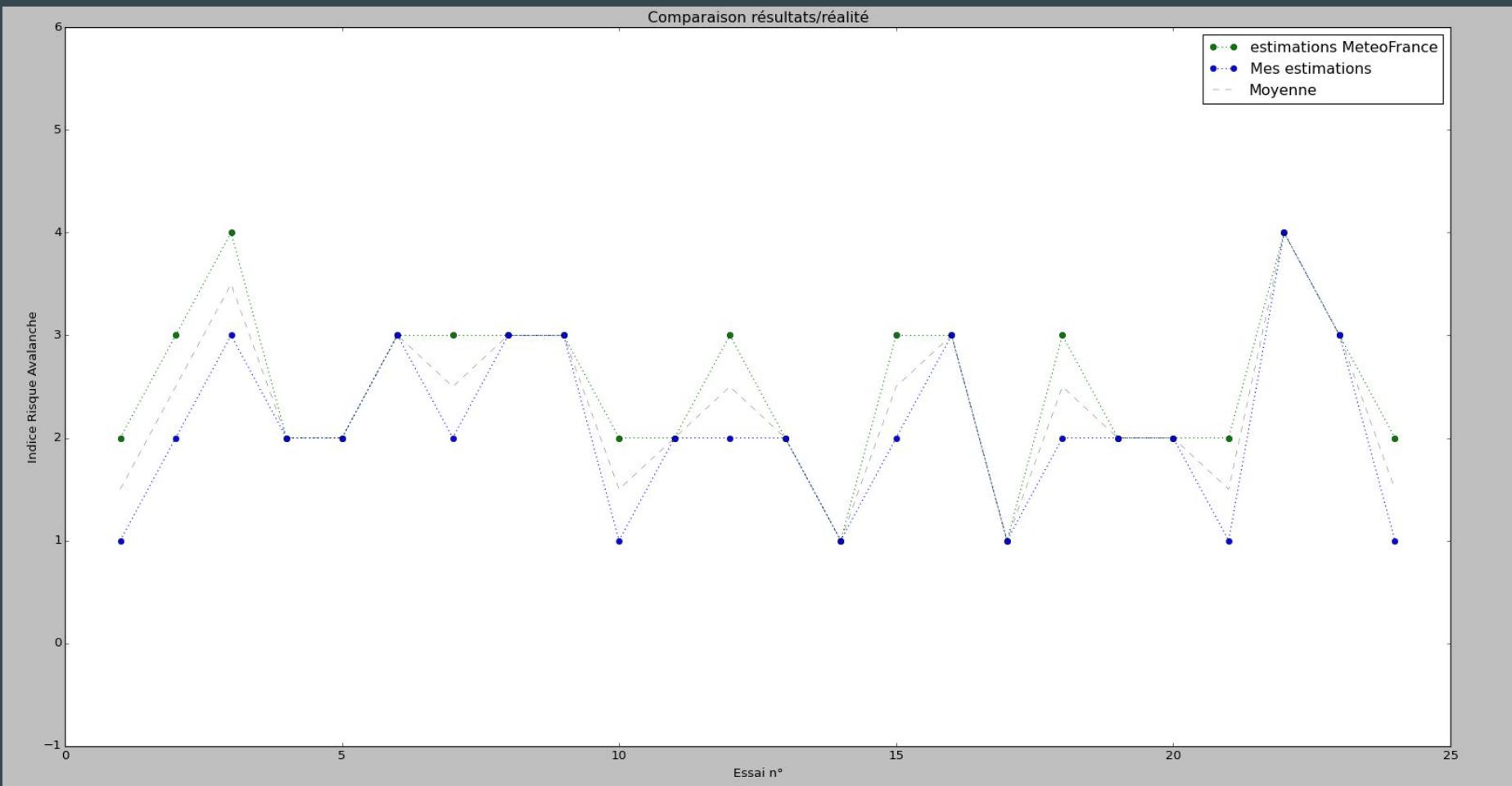


Rédigé par Météo-France avec la contribution des observateurs du réseau nivo-météorologique. Partenariat : ANMSM (Maires de Stations de Montagne), DSF (Domaines Skiables de France), ADSP (Directeurs de Pistes et de la Sécurité des Stations de Sports d'Hiver) et autres acteurs de la montagne.

3.3 - Algorithme de comparaison de mes estimations avec celles de MétéoFrance.

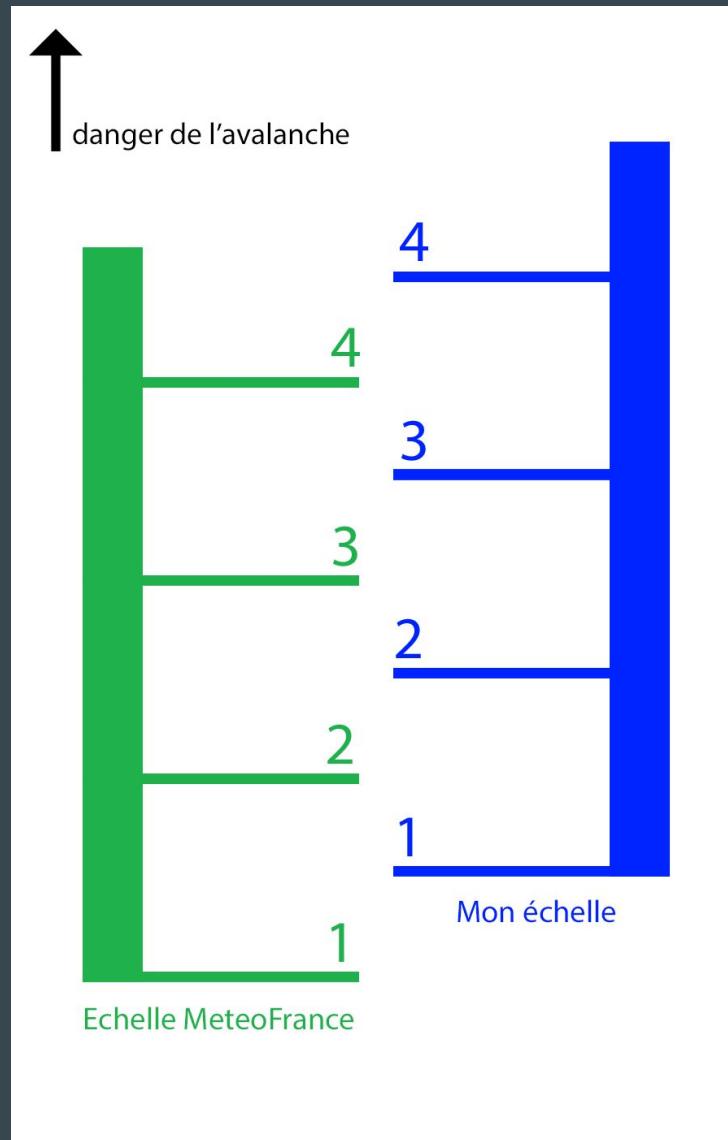
```
#28/12/2016 | Chablais :  
L14 = [ 1 , [ "Soleil" ,0 , 10 , 1 , 50 ] , [" Soleil" , 0 , 20 , 2,50 ] , [ "Soleil" , -1 , 30 , 1 ,50 ] , ["Soleil" ,0 , 10 , 1 ,50 ] , ["Soleil" ,-1 , 50 , 1 ,50 ] , ["Soleil" ,0 , 50 , 1 ,50 ] ]  
  
#06/02/2019 | Vercors :  
L15 = [ 3 , [ "Pluie" , -2 , 80 , 0 , 130 ] , [" Pluie" , -2 , 20 , 0,150 ] , [ "Pluie" , -2 , 70 , 0 ,190 ] , ["Soleil" ,-2 , 30 , 1 ,210 ] , ["Soleil" ,0 , 20 , 1 ,205 ] , ["Soleil" ,1 , 30 , 1 ,200 ] ]
```

3.3 - Algorithme de comparaison de mes estimations avec celles de MétéoFrance.



```
In [44]: CompareMeteo()  
Out[44]: 0.4166666666666667
```

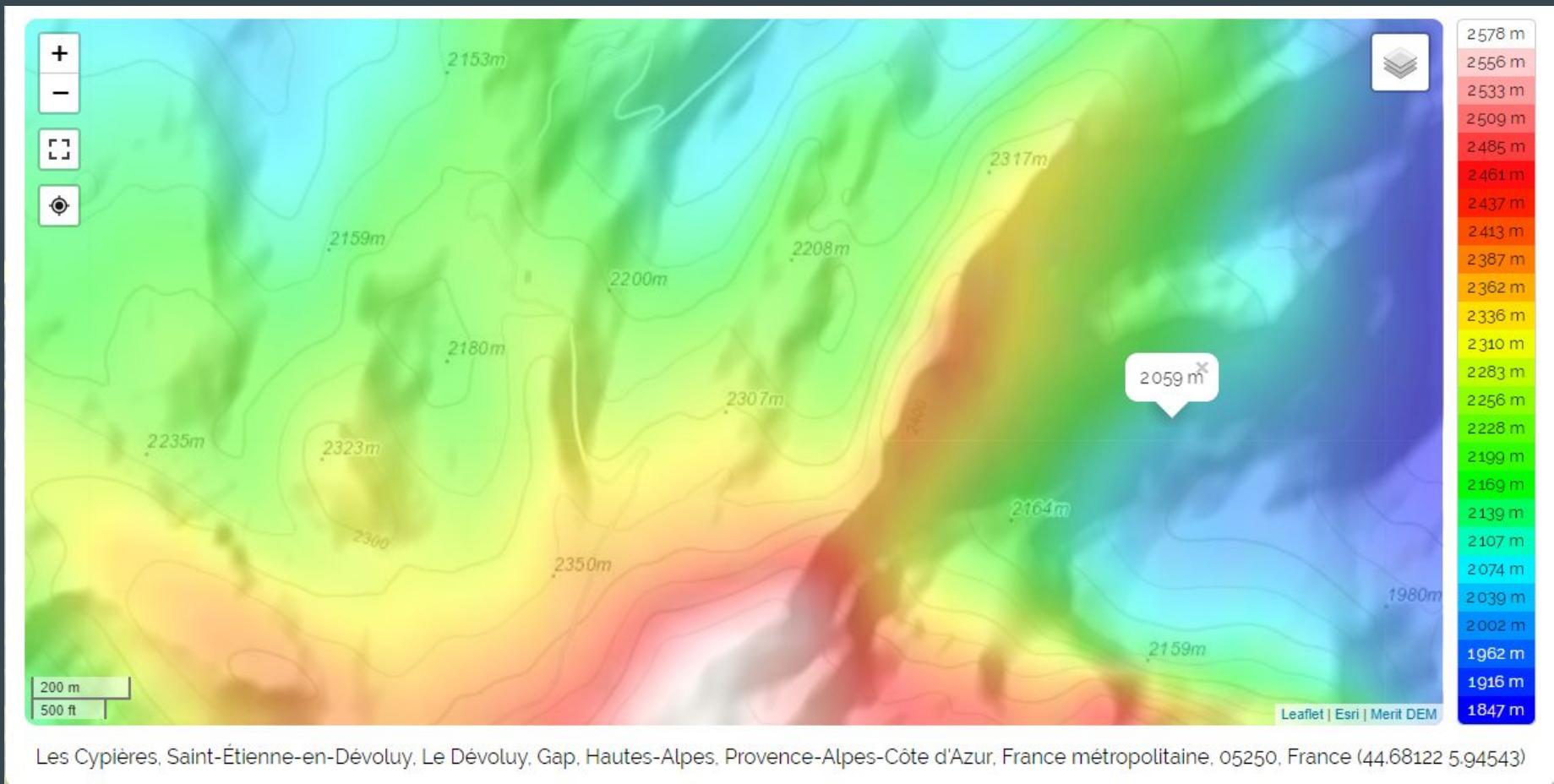
3.3 - Algorithme de comparaison de mes estimations avec celles de MétéoFrance.



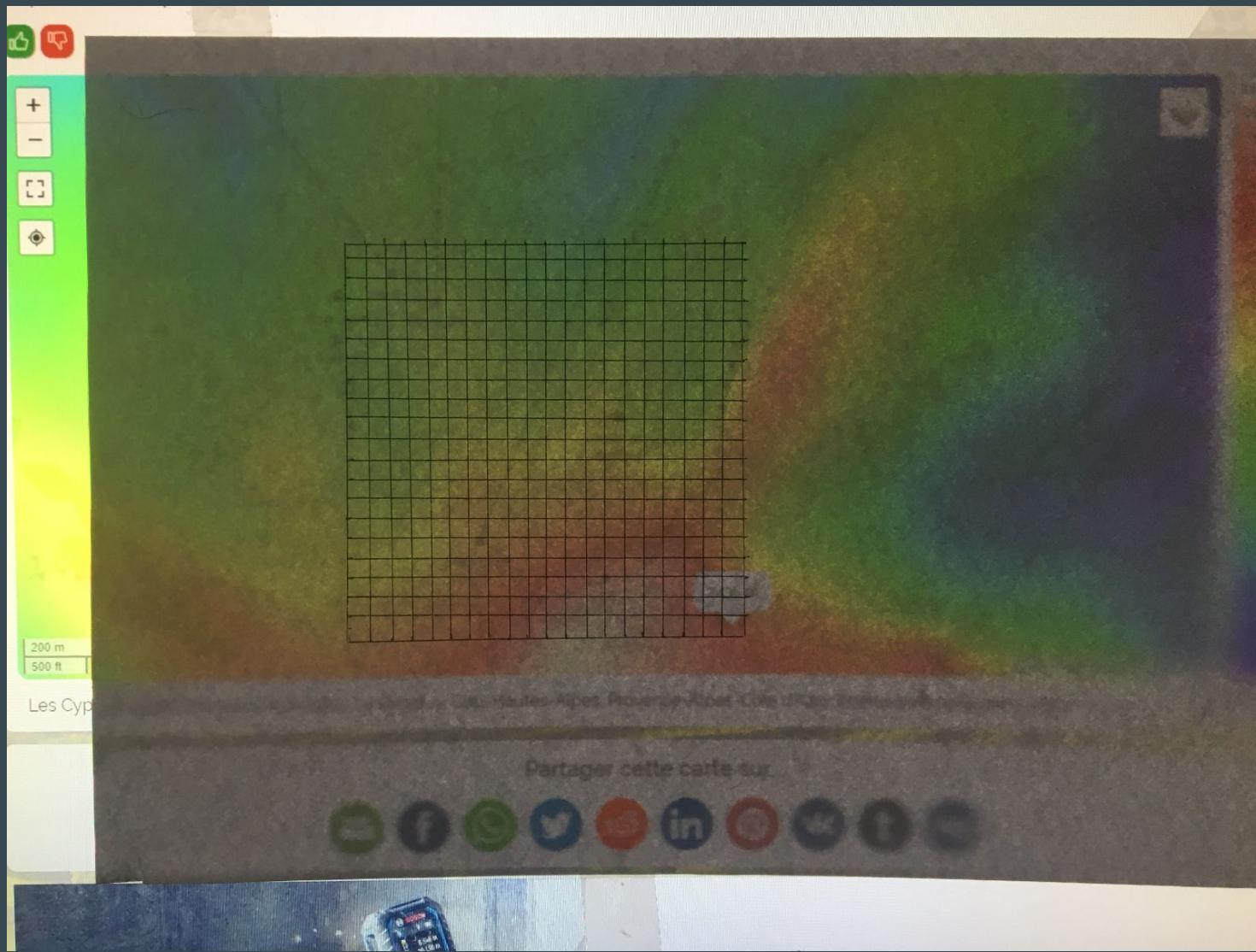
3.4 - Génération d'une matrice d'altitude d'une montagne.

100	110	120
100	110	110
100	100	120

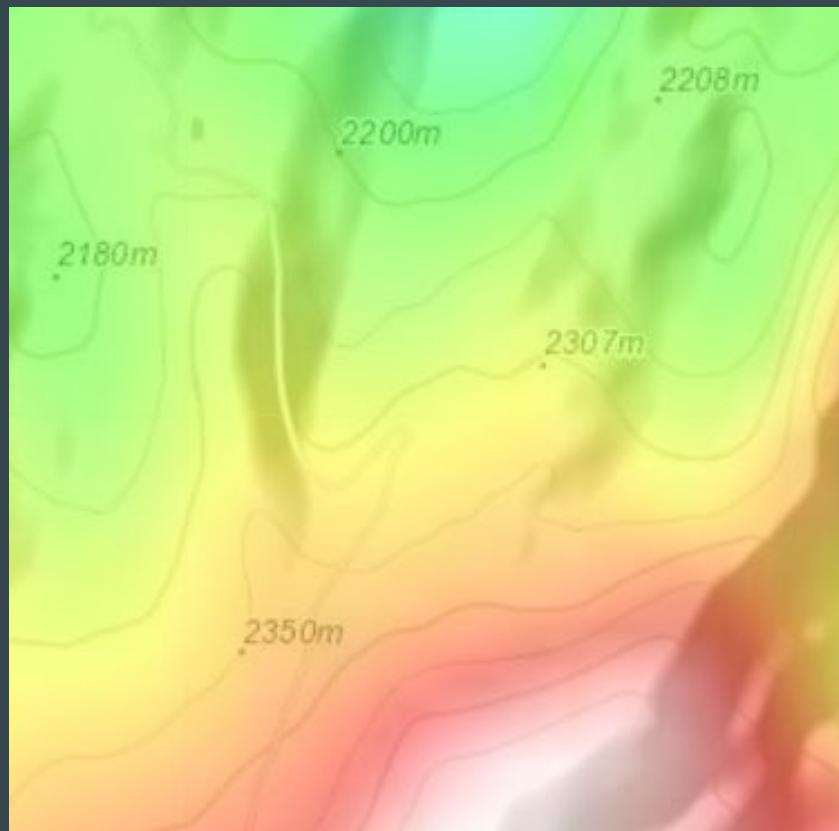
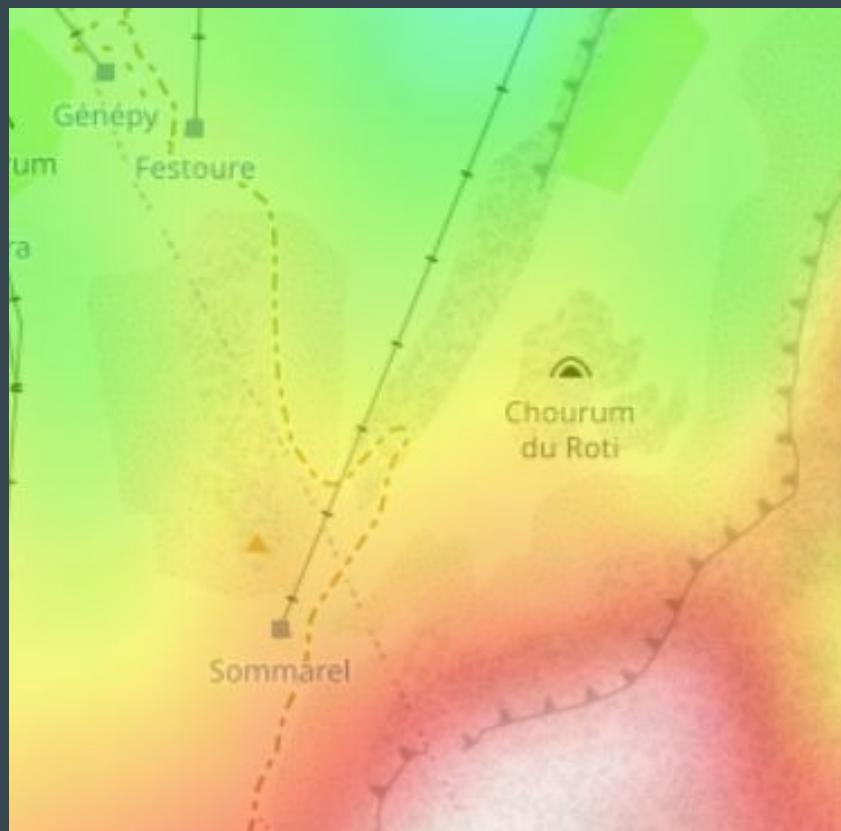
3.4 - Génération d'une matrice d'altitude d'une montagne.



3.4 - Génération d'une matrice d'altitude d'une montagne.



3.4 - Génération d'une matrice d'altitude d'une montagne.



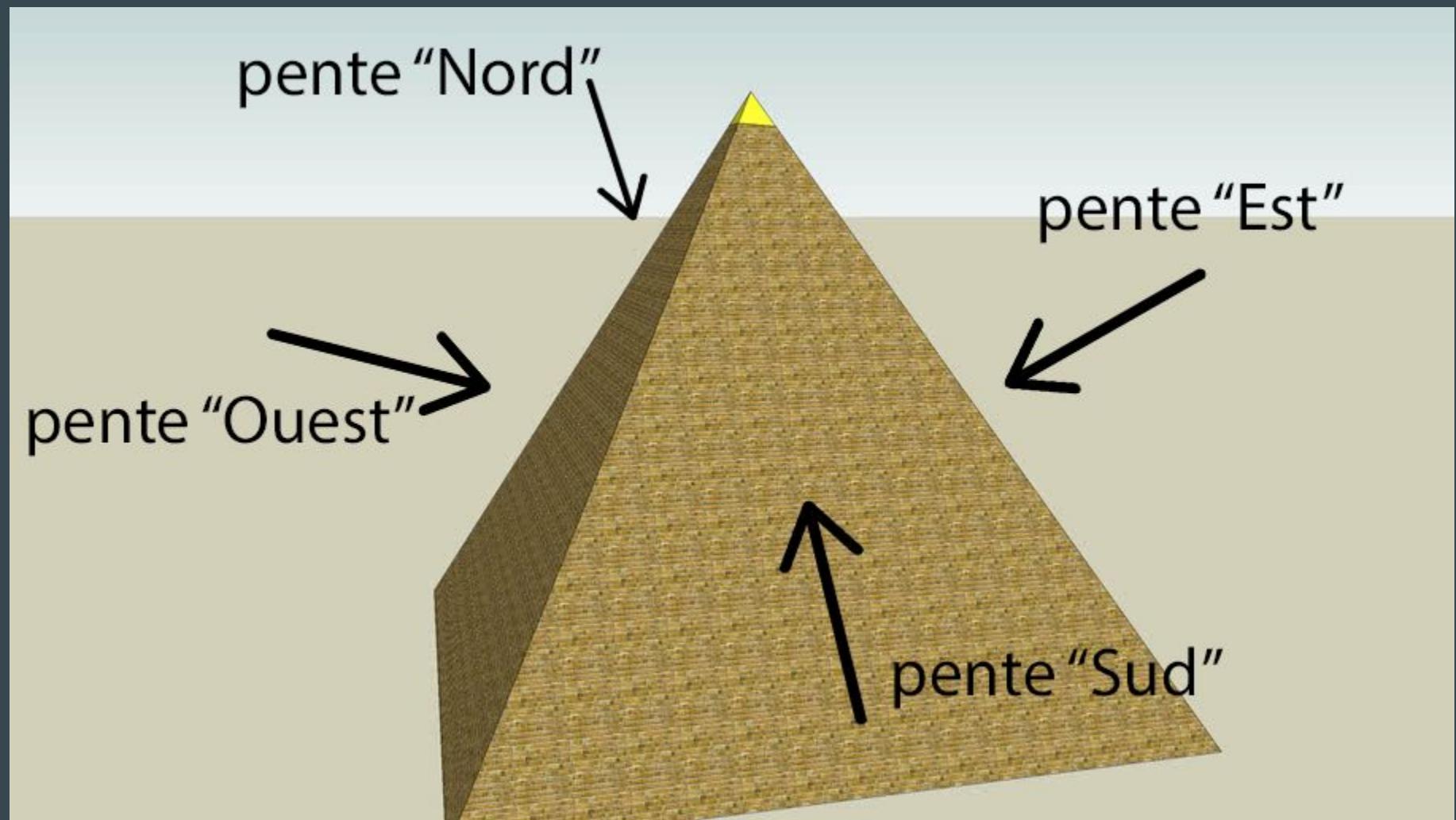
3.4 - Génération d'une matrice d'altitude d'une montagne.

```
1  
2  
3  
4 M = [  
5 [2208 ,2211 ,2215 ,2215 ,2203 ,2192 ,2180 ,2171 ,2151 ,2132 ,2119 ,2117 ,2127 ,2149 ,2181 ,2197 ,2203 ,2200 ,2193 ,2190 ],  
6 [2208 ,2214 ,2219 ,2222 ,2213 ,2204 ,2193 ,2182 ,2162 ,2147 ,2134 ,2135 ,2147 ,2168 ,2197 ,2204 ,2206 ,2202 ,2197 ,2190 ],  
7 [2208 ,2215 ,2223 ,2229 ,2226 ,2221 ,2210 ,2200 ,2175 ,2462 ,2154 ,2157 ,2168 ,2184 ,2207 ,2208 ,2208 ,2205 ,2201 ,2206 ],  
8 [2204 ,2213 ,2225 ,2236 ,2237 ,2231 ,2223 ,2210 ,2192 ,2182 ,2175 ,2182 ,2195 ,2206 ,2218 ,2214 ,2210 ,2206 ,2205 ,2211 ],  
9 [2197 ,2210 ,2225 ,2242 ,2246 ,2243 ,2234 ,2220 ,2206 ,2198 ,2196 ,2202 ,2215 ,2225 ,2226 ,2221 ,2212 ,2206 ,2209 ,2217 ],  
10 [2192 ,2207 ,2224 ,2243 ,2258 ,2255 ,2245 ,2232 ,2219 ,2215 ,2218 ,2226 ,2237 ,2240 ,2236 ,2226 ,2215 ,2209 ,2215 ,2248 ],  
11 [2188 ,2201 ,2219 ,2244 ,2269 ,2267 ,2256 ,2241 ,2231 ,2233 ,2238 ,2248 ,2258 ,2260 ,2245 ,2230 ,2218 ,2213 ,2223 ,2269 ],  
12 [2195 ,2202 ,2222 ,2243 ,2268 ,2277 ,2274 ,2253 ,2250 ,2252 ,2259 ,2270 ,2275 ,2272 ,2260 ,2243 ,2232 ,2232 ,2240 ,2300 ],  
13 [2206 ,2208 ,2222 ,2242 ,2267 ,2286 ,2292 ,2267 ,2266 ,2274 ,2282 ,2287 ,2292 ,2287 ,2273 ,2256 ,2247 ,2246 ,2259 ,2325 ],  
14 [2220 ,2218 ,2229 ,2249 ,2274 ,2294 ,2302 ,2284 ,2285 ,2292 ,2300 ,2307 ,2310 ,2309 ,2293 ,2277 ,2267 ,2372 ,2304 ,2353 ],  
15 [2237 ,2231 ,2241 ,2260 ,2287 ,2307 ,2313 ,2306 ,2307 ,2313 ,2319 ,2325 ,2332 ,2333 ,2321 ,2301 ,2293 ,2308 ,2348 ,2369 ],  
16 [2253 ,2248 ,2256 ,2278 ,2304 ,2302 ,2331 ,2325 ,2323 ,2325 ,2335 ,2339 ,2344 ,2342 ,2338 ,2324 ,2321 ,2330 ,2361 ,2363 ],  
17 [2266 ,2261 ,2270 ,2292 ,2317 ,2334 ,2346 ,2342 ,2337 ,2341 ,2348 ,2353 ,2356 ,2357 ,2356 ,2355 ,2355 ,2356 ,2368 ,2356 ],  
18 [2277 ,2275 ,2282 ,2299 ,2324 ,2344 ,2355 ,2353 ,2353 ,2361 ,2366 ,2371 ,2374 ,2377 ,2386 ,2396 ,2399 ,2390 ,2373 ,2347 ],  
19 [2290 ,2288 ,2296 ,2309 ,2333 ,2351 ,2364 ,2365 ,2369 ,2377 ,2387 ,2389 ,2396 ,2399 ,2418 ,2436 ,2437 ,2424 ,2381 ,2339 ],  
20 [2304 ,2307 ,2311 ,2322 ,2342 ,2358 ,2371 ,2381 ,2393 ,2403 ,2419 ,2433 ,2443 ,2464 ,2476 ,2484 ,2475 ,2449 ,2399 ,2353 ],  
21 [2320 ,2322 ,2327 ,2334 ,2351 ,2366 ,2381 ,2398 ,2416 ,2435 ,2449 ,2466 ,2492 ,2510 ,2427 ,2523 ,2501 ,2467 ,2415 ,2371 ],  
22 [2333 ,2337 ,2343 ,2350 ,2365 ,2380 ,2396 ,2415 ,2439 ,2465 ,2491 ,2517 ,2532 ,2545 ,2551 ,2537 ,2515 ,2477 ,2433 ,2398 ],  
23 [2347 ,2355 ,2361 ,2369 ,2381 ,2395 ,2412 ,2432 ,2465 ,2496 ,2527 ,2555 ,2566 ,2567 ,2564 ,2546 ,2522 ,2489 ,2453 ,2423 ],  
24 [2365 ,2374 ,2384 ,2392 ,2405 ,2415 ,2431 ,2457 ,2494 ,2522 ,2552 ,2569 ,2570 ,2563 ,2554 ,2538 ,2521 ,2493 ,2462 ,2436 ]  
25 ]
```

3.5 - Algorithme de détermination des pentes d'une matrice d'altitude.

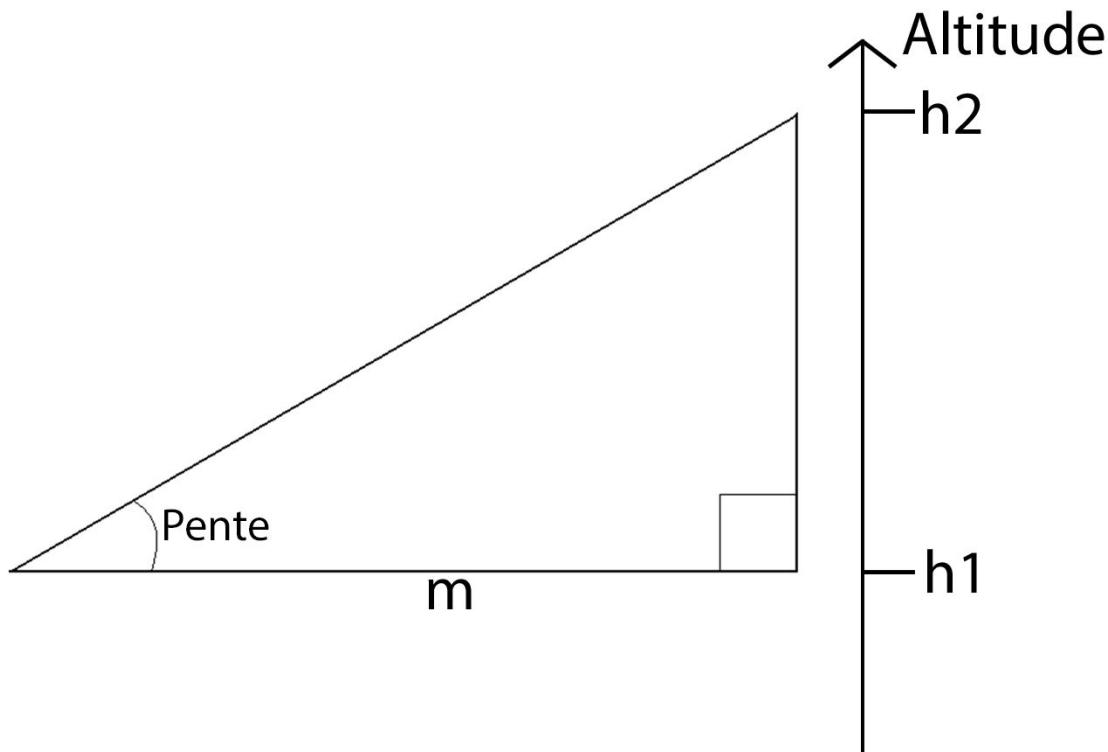
100	110	120	100	1210	120
0	0	12	100	12100	10
0	12	12	100	100	22
100	100	120	100	100	22

3.5 - Algorithme de détermination des pentes d'une matrice d'altitude.



3.5 - Algorithme de détermination des pentes d'une matrice d'altitude.

$$\text{pente} = (\arctan((h_2-h_1) / m)) * 180 / \pi$$



3.5 - Algorithme de détermination des pentes d'une matrice d'altitude.

```
[ 'Ouest', '14.574216198038739'],
[ 'Ouest', '6.84277341263094'],
[ 'Est', '15.642246457208728'],
[ 'Est', '23.749494492866766'],
[ 'Est', '37.234833981574674'],
[ 'Est', '41.34777721969366'],
[ 'Est', '34.992020198558656']],
```

```
[[ 'Ouest', '9.090276920822323'],
[ 'Ouest', '6.84277341263094'],
[ 'Ouest', '9.090276920822323'],
[ 'Ouest', '13.495733280795813'],
[ 'Ouest', '15.642246457208728'],
[ 'Ouest', '18.778033222445544'],
[ 'Ouest', '21.80140948635181'],
[ 'Ouest', '33.424811182603804'],
[ 'Ouest', '31.798912824294423'],
[ 'Ouest', '31.798912824294423'],
[ 'Ouest', '29.248826336546976'],
[ 'Ouest', '12.407418527400743'],
[ 'Ouest', '1.1457628381751035'],
[ 'Est', '3.4336303624505224'],
[ 'Est', '19.79887635452493'],
[ 'Est', '25.64100582430528'],
[ 'Est', '33.424811182603804'],
[ 'Est', '35.75388725443675'],
[ 'Est', '30.96375653207352']],
```

```
[[ 'Ouest', '10.203973721731684'],
[ 'Ouest', '11.309932474020213'],
[ 'Ouest', '9.090276920822323'],
[ 'Ouest', '14.574216198038739'],
[ 'Ouest', '11.309932474020213'],
[ 'Ouest', '17.744671625056935'],
[ 'Ouest', '27.474431626277134'],
[ 'Ouest', '36.50144112050632'],
[ 'Ouest', '29.248826336546976'],
[ 'Ouest', '30.96375653207352']]
```

```
[ 'Nord', '34.992020198558656'],
[ 'Nord', '68.03943597990354'],
[ 'Nord', '15.642246457208728'],
[ 'Nord', '15.642246457208728'],
[ 'Nord', '11.309932474020213'],
[ 'Nord', '19.79887635452493'],
[ 'Nord', '28.369046293278583']],
```

```
[[ 'Nord', '15.642246457208728'],
[ 'Nord', '19.79887635452493'],
[ 'Nord', '19.79887635452493'],
[ 'Nord', '20.80679101271123'],
[ 'Nord', '17.744671625056935'],
[ 'Nord', '16.69924423399362'],
[ 'Nord', '17.744671625056935'],
[ 'Nord', '18.778033222445544'],
[ 'Nord', '27.474431626277134'],
[ 'Nord', '31.798912824294423'],
[ 'Nord', '35.75388725443675'],
[ 'Nord', '37.234833981574674'],
[ 'Nord', '34.21570213243741'],
[ 'Nord', '23.749494492866766'],
[ 'Nord', '14.574216198038739'],
[ 'Nord', '10.203973721731684'],
[ 'Nord', '7.96961039432136'],
[ 'Nord', '13.495733280795813'],
[ 'Nord', '21.80140948635181'],
[ 'Nord', '26.56505117707799']],
```

```
[[ 'Nord', '19.79887635452493'],
[ 'Nord', '20.80679101271123'],
[ 'Nord', '24.702430227771313'],
[ 'Nord', '24.702430227771313'],
[ 'Nord', '25.64100582430528'],
[ 'Nord', '21.80140948635181'],
[ 'Nord', '20.80679101271123'],
[ 'Nord', '26.56505117707799'],
[ 'Nord', '30.112733150082427']]
```

3.6 - Algorithme de détermination des pentes à risques d'une montagne en fonction du facteur risque avalanche et du bulletin météo.

```
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 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, 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, 2],  
 [0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],  
 [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],  
 [0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],  
 [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2],  
 [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],  
 [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],  
 [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],  
 [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 2, 0, 2, 2],  
 [0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2],  
 [0, 0, 0, 0, 0, 0, 2, 2, 2, 1, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2],  
 [0, 0, 0, 0, 0, 1, 2, 1, 2, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 1]]
```

Pentes horizontales

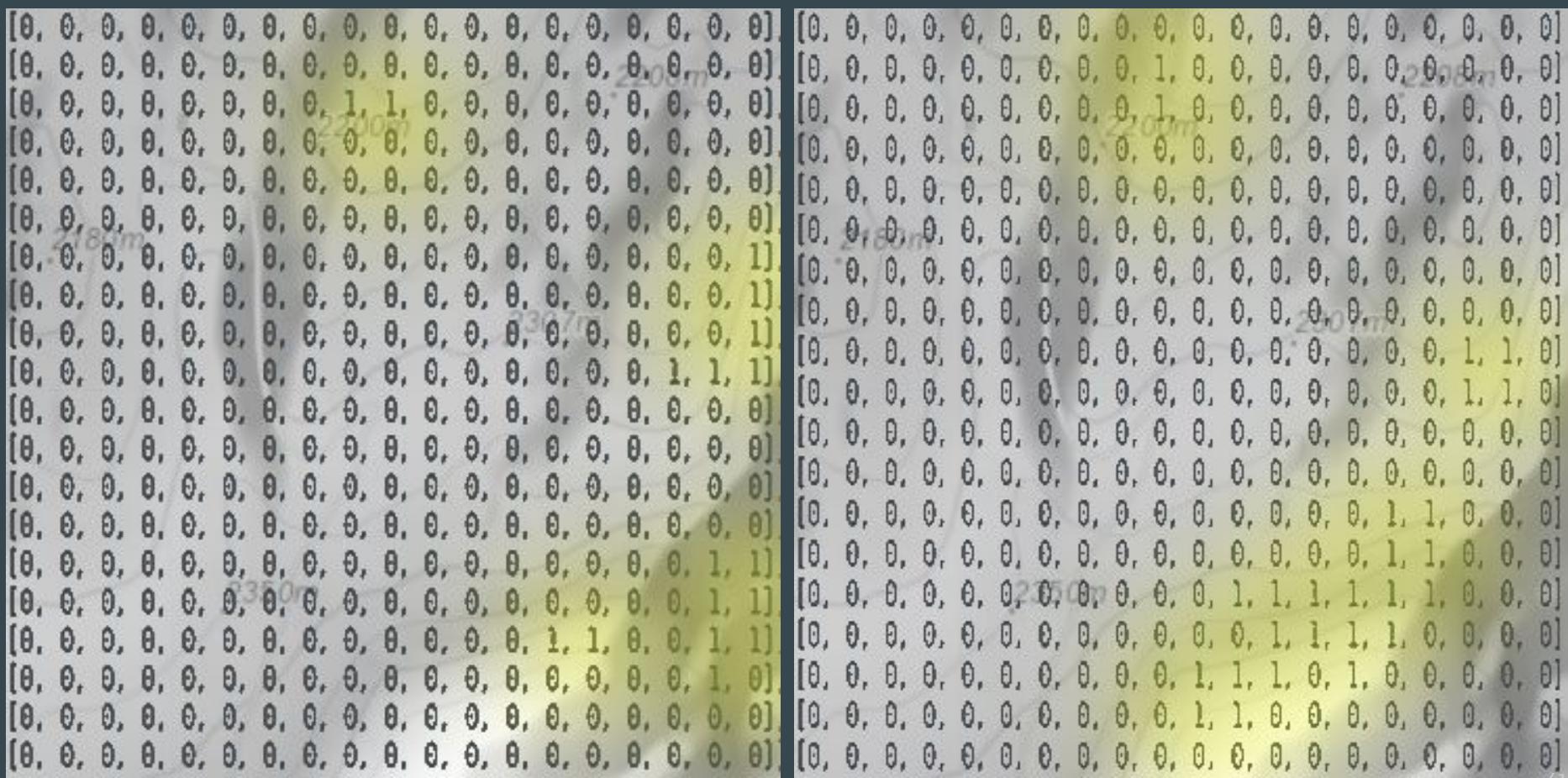
4.6 - Algorithme de détermination des pentes à risques d'une montagne en fonction du facteur risque avalanche et du bulletin météo.

```
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 2],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 2],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 2, 2, 2, 2],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 2, 2, 2, 2, 2, 2, 2, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 2, 2, 2, 2, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 2, 2, 2, 2, 0, 0, 0],  
 [0, 1, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 2],  
 [0, 0, 0, 1, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 1, 0, 0, 0, 0, 0, 0, 1, 2],  
 [0, 1, 1, 1, 2, 1, 1, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]]
```

Pentes verticales

3.6 - Algorithme de détermination des pentes à risques d'une montagne en fonction du facteur risque avalanche et du bulletin météo.

Risque 1 - 40 km/h - Sud - Pentes horizontales | Pentes verticales



3.6 - Algorithme de détermination des pentes à risques d'une montagne en fonction du facteur risque avalanche et du bulletin météo.

Risque 2 - pas de vent - Pentes horizontales | Pentes verticales



3.6 - Algorithme de détermination des pentes à risques d'une montagne en fonction du facteur risque avalanche et du bulletin météo.

Risque 4 - 90 km/h - Est - Pentes horizontales | Pentes verticales

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 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, 2, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 2, 2, 1, 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, 1, 1, 1, 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, 1, 2, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1]
[0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 2]
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 2]
[0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 2]
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 2, 2, 2]
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 2, 2, 2, 2, 0]
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 2, 1, 0, 0]
[0, 0, 0, 1, 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, 2, 2, 0]
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 2, 2, 2, 0]
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 2, 2, 2, 2, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 2, 2, 2, 2, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 2, 2, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 2, 2, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 2, 2, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 1, 0, 0, 0, 0, 0, 1, 2, 2, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 2]
[0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 2, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1]	[0, 1, 1, 1, 2, 1, 1, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

4 - Bilan et limites

```
1 #Le but de cet algorithme est de générer un bulletin météo réaliste.
2 #Pour cela il générera un 5-uplet pour 6 jours consécutifs : [Temps(Soleil,Pluie) , Température(°C), Vent(km/h), Sens du vent{0,1,2,3}), Enneigement(cm)].
3 #Pour notre étude nous supposerons le temps, la température, le vent, son sens et l'enneigement uniforme sur toute la zone étudiée.
4 #On supposera que le vent ne souffle que dans une seule direction à la fois.
5 def GenererBulletinMeteo () :
6     from random import randint
7     Bulletin = []
8     #Initialisation jour 0 :
9     Te = 0
10    TC = 0
11    V = 0
12    dV= 0
13    E = 0
14
15    if randint(0,1) == 1 :           #Probabilité de 1 chance sur 2
16        Te = "Pluie"
17    else :
18        Te = "Soleil"
19    if Te == "Pluie" :
20        TC = -6 + randint(-3,3)
21    else :
22        TC = 9 + randint(-3,3)
23    if Te == "Pluie" :
24        V = 30 + randint(0,20)
25    if Te == "Pluie" :
26        dV = randint(0,3)
27    E = 125 + randint(-50,50)
28
29    Bulletin.append([Te,TC,V,dV,E])
30
31
32    for k in range (5) :
33        Te, TC, V, dV, E = 0,0,0,0,0
34        #Temps qu'il fait
35        if Bulletin[k][0] == "Pluie" :
36            if randint(0,1) == 0 :      #<-- 1 chance sur 2
37                Te = "Pluie"
38            else :
39                Te = "Soleil"
40
41            if randint(0,7) == 0:      #<-- 1 chance sur 8
42                Te = "Pluie"
43            else :
44                Te = "Soleil"
45
46        #Température
47        if Bulletin[k][0] == "Pluie" :
48            if Te == "Pluie" :
49                TC = Bulletin[k][1] + randint(-5,5)
50            if Te == "Soleil" :
51                TC = Bulletin[k][1] + randint(-5,5) + 15
52        else :
53            if Te == "Pluie" :
54                TC = Bulletin[k][1] + randint(-5,5) - 15
55            if Te == "Soleil" :
56                TC = Bulletin[k][1] + randint(-5,5)
57
58        #Vent
59        if Te == "Pluie" :
60            V = 30 + randint(0,20)
61
62        #Sens du vent
63        if Bulletin[k][0] == "Pluie" :
64            if randint(0,1) == 0 :
65                dV = Bulletin[k][3]
66            else :
67                dV = randint(0,3)
68
69        #Enneigement
70
71        if Te == "Pluie" :
72            E = Bulletin[k][4] + randint(-3,3) + 10
73        else :
74            E = Bulletin[k][4] + randint(-2,2) - 5
75
76    Bulletin.append([Te,TC,V,dV,E])
77
78 return Bulletin
```

```
1 def IndiceRisqueAvalanche(L) :
2
3     I = 0
4     IV = 0
5     IE = 0
6     IM = 0
7     B = L
8
9     #On considère le jour J comme le jour 6 du bulletin (position 5)
10    #1 : Indice de risque lié au vent :
11    if B[5][2] < 30 :
12        IV = 1
13    if B[5][2] > 30 and B[5][3] < 40 :
14        IV = 2
15    if B[5][2] > 40 and B[5][3] < 50 :
16        IV = 3
17    if B[5][2] > 50 :
18        IV = 4
19
20    #2 : Indice de risque lié à l'enneigement :
21    if B[5][4] < 50 :
22        IE = 1
23    if B[5][4] > 50 and B[5][4] < 100 :
24        IE = 2
25    if B[5][4] > 100 and B[5][4] < 150 :
26        IE = 3
27    if B[5][4] > 150 :
28        IE = 4
29
30    #3 : Indice de risque lié à la météo :
31    #3.1 : Patterns risques 4 :
32    if B[5][0] == "Soleil" and B[4][0] == "Soleil" and B[3][0] == "Pluie" and B[3][1]<0 and B[2][0] == "Soleil" and B[1][0] == "Soleil" :
33        IM = 4
34    elif B[4][0] == "Soleil" and B[3][0] == "Soleil" and B[2][0] == "Pluie" and B[2][1]<0 and B[1][0] == "Soleil" and B[0][0] == "Soleil" :
35        IM = 4
36    elif B[5][0] == "Soleil" and B[4][2] > 30 and B[3][1] < 0 and B[2][1] < 0 and B[1][0] == "Pluie" and B[1][1] < 0 :
37        IM = 4
38    elif B[5][0] == "Pluie" and B[5][1] > 0 and B[4][0] == "Pluie" and B[4][1] > 0 and B[3][0] == "Pluie" and B[3][1] < 0 and B[2][1] < 0 :
39        IM = 4
40
41    #3.2 : Patterns de risque 3 :
42    if IM == 0 :
43        for k in range (2) :
44            if B[k][0] == "Soleil" and B[k+1][0] == "Soleil" and B[k+2][0] == "Pluie" and B[k+2][1] < 0 and B[k+3][0] == "Soleil" :
45                IM = 3
46        for k in range (2) :
47            if B[k][0] == "Soleil" and B[k+2][0] == "Soleil" and B[k+1][0] == "Pluie" and B[k+1][1] < 0 and B[k+3][0] == "Soleil" :
48                IM = 3
49        if B[5][0] == "Soleil" and B[4][2] > 30 and B[3][2] < 0 and B[2][0] == "Pluie" and B[2][1] < 0 :
50            IM = 3
51        elif B[5][0] == "Soleil" and B[4][2] > 30 and B[3][0] == "Pluie" and B[3][1] < 0 :
52            IM = 3
53        elif B[5][2] > 30 and B[4][1] < 0 and B[3][1] < 0 and B[2][1] < 0 and B[2][0] == "Pluie" :
54            IM = 3
55
56    #3.3 : Patterns de risque 2:
57    if IM == 0 :
58        for k in range (3) :
59            if B[k][0] == "Soleil" and B[k+1][0] == "Pluie" and B[k+1][1] < 0 and B[k+2][0] == "Soleil" :
60                IM = 2
61        for k in range (4) :
62            if B[k][1]<0 and B[k+1][2] > 30 :
63                IM = 2
64            elif B[k][2] > 30 and B[k+1][0] == "Soleil" :
65                IM = 2
66            elif B[k+1][0] == "Pluie" and B[k+1][1]<0 and B[k][1] < 0 :
67                IM = 2
68
69    #3.4 : Patterns de risque 1 :
70
71    if IM == 0 :
72        IM = 1
73
74    I = round( ( (IM + IV + IE ) / 3) , 0)
75
76
77
78
79
80
return( int(I) )
```

A
N
N
E
X

3

```
1 def PourcentageIndices (n) :
2     L = [0,0,0,0]
3     for k in range (n) :
4         B = GenereBulletinMeteo()
5         L [ int( IndiceRisqueAvalanche( B ) - 1 ) ] += 1
6     for k in range (len(L)) :
7         L[k] = (100 * L[k]) / n
8     return L
```

```

1 def DataMeteoListe () :
2     # Nord Sud Est Ouest
3     L=[]
4     #21/12/2021 | Chablais [ RisqueJJ ] , [J-6], [J-4], [J-3], [J-2], [J-1], [J]
5     L1 = [ 2 , [ "Soleil" ,1 , 50 , 1 , 100 ] , [ "Soleil" , 0.25 , 3 , 100 ] , [ "Soleil" , 1.30 , 3 , 100 ] , [ "Soleil" ,-3 , 30 , 1 , 90 ] , [ "Soleil" ,-3 , 50 , 1 , 85 ] , [ "Soleil" , -5 , 30 , 2 , 85 ] ]
6     #06/01/2022 | Chablais
7     L2 = [ 3 , [ "Soleil" ,1 , 10 , 2 , 60 ] , [ "Soleil" , 1 , 50 , 0 , 50 ] , [ "Pluie" , -2.45 , 0 , 40 ] , [ "Pluie" ,1 , 80 , 0 , 40 ] , [ "Pluie" , -3 , 40 , 1 , 55 ] , [ "Soleil" , -5 , 50 , 0 , 55 ] ]
8     #07/02/2022 | Chablais
9     L3 = [ 4 , [ "Pluie" , -10 , 40 , 1 , 80 ] , [ "Soleil" , -8 , 10 , 0 , 80 ] , [ "Soleil" , 2.50 , 0 , 80 ] , [ "Pluie" , -4 , 20 , 2 , 80 ] , [ "Soleil" , -3 , 40 , 0 , 80 ] , [ "Pluie" , -3 , 50 , 1 , 90 ] ]
10    #14/01/2022 | Devoluy
11    L4 = [ 2 , [ "Pluie" ,3 , 90 , 1 , 65 ] , [ "Soleil" , 3.80 , 1 , 75 ] , [ "Soleil" ,3.30 , 3 , 70 ] , [ "Soleil" ,1 , 30 , 3 , 65 ] , [ "Soleil" , -2 , 40 , 3 , 65 ] , [ "Soleil" , -2 , 40 , 3 , 60 ] ]
12    #02/02/2022 | Chartreuse
13    L5 = [ 2 , [ "Soleil" ,1 , 30 , 1 , 80 ] , [ "Soleil" , -1.23 , 1 , 80 ] , [ "Soleil" ,1.25 , 1 , 80 ] , [ "Pluie" , -3 , 25 , 1 , 80 ] , [ "Pluie" , -3 , 50 , 1 , 80 ] , [ "Pluie" , -1 , 40 , 1 , 80 ] ]
14    #13/01/2022 | Aravis
15    L6 = [ 3 , [ "Pluie" ,1 , 0 , 0 , 45 ] , [ "Pluie" ,1.53 , 2 , 70 ] , [ "Pluie" ,1 , 50 , 3 , 100 ] , [ "Pluie" ,1 , 40 , 1 , 110 ] , [ "Soleil" ,0 , 50 , 3 , 105 ] , [ "Soleil" , -1 , 50 , 3 , 105 ] ]
16    L.append (L1)
17    L.append (L2)
18    L.append (L3)
19    L.append (L4)
20    L.append (L5)
21    L.append (L6)
22
23
24
25
26
27
28
29
30     #10/02/2021 | Chablais :
31     L7 = [ 3 , [ "Pluie" , -6 , 60 , 0 , 80 ] , [ "Soleil" , -6 , 73 , 0 , 80 ] , [ "Pluie" , -6 , 50 , 0 , 70 ] , [ "Soleil" , -2 , 40 , 0 , 70 ] , [ "Pluie" , -2 , 43 , 0 , 70 ] , [ "Pluie" , -2 , 40 , 2 , 70 ] ]
32
33     #29/12/2020 | Mont-Blanc :
34     L8 = [ 3 , [ "Pluie" , -2 , 70 , 0 , 40 ] , [ "Pluie" , -10 , 60 , 1 , 60 ] , [ "Soleil" , -10 , 60 , 0 , 60 ] , [ "Pluie" , -10 , 80 , 0 , 70 ] , [ "Pluie" , -10 , 90 , 0 , 70 ] ]
35
36     #10/02/2021 | Vanoise :
37     L9 = [ 3 , [ "Soleil" ,1 , 80 , 0 , 110 ] , [ "Soleil" , 1 , 80 , 1 , 110 ] , [ "Pluie" , -1 , 80 , 0 , 110 ] , [ "Soleil" , -1 , 70 , 1 , 110 ] , [ "Pluie" , -1 , 60 , 2 , 115 ] , [ "Pluie" , -1 , 50 , 2 , 120 ] ]
38
39     #11/01/2019 | Bauges :
40     L10 = [ 2 , [ "Soleil" ,1 , 30 , 1 , 40 ] , [ "Soleil" , 1 , 40 , 1 , 30 ] , [ "Soleil" , -2 , 30 , 1 , 30 ] , [ "Pluie" , -3 , 30 , 1 , 40 ] , [ "Pluie" , -4 , 30 , 1 , 40 ] , [ "Soleil" , -5 , 30 , 1 , 40 ] ]
41
42     #07/02/2018 | Devoluy :
43     L11 = [ 2 , [ "Soleil" , -1 , 20 , 1 , 40 ] , [ "Soleil" , -10 , 60 , 1 , 30 ] , [ "Soleil" , -2 , 30 , 1 , 30 ] , [ "Pluie" , -2 , 60 , 0 , 40 ] , [ "Soleil" , -2 , 40 , 3 , 40 ] , [ "Soleil" , -2 , 10 , 1 , 40 ] ]
44
45     #23/12/2017 | Aravis :
46     L12 = [ 3 , [ "Pluie" , -4 , 30 , 1 , 130 ] , [ "Soleil" , -3 , 40 , 1 , 130 ] , [ "Soleil" , -2 , 50 , 1 , 125 ] , [ "Soleil" , -1 , 50 , 1 , 125 ] , [ "Soleil" , -0 , 40 , 1 , 125 ] , [ "Pluie" ,2 , 30 , 1 , 120 ] ]
47     L.append (L7)
48     L.append (L8)
49     L.append (L9)
50     L.append (L10)
51     L.append (L11)
52     L.append (L12)
53
54
55     #06/01/2022 | Haute-Bigorre :
56     L13 = [ 2 , [ "Soleil" ,2 , 20 , 2 , 110 ] , [ "Soleil" , 2 , 40 , 2 , 110 ] , [ "Soleil" , 2 , 50 , 2 , 100 ] , [ "Soleil" ,2 , 90 , 2 , 100 ] , [ "Pluie" ,0 , 40 , 2 , 120 ] , [ "Soleil" , -4 , 40 , 1 , 120 ] ]
57
58     #28/12/2016 | Chablais :
59     L14 = [ 1 , [ "Soleil" ,0 , 10 , 1 , 50 ] , [ "Soleil" , 0 , 20 , 2 , 50 ] , [ "Soleil" , -1 , 30 , 1 , 50 ] , [ "Soleil" ,0 , 10 , 1 , 50 ] , [ "Soleil" , -1 , 50 , 1 , 50 ] , [ "Soleil" ,0 , 50 , 1 , 50 ] ]
60
61     #06/02/2019 | Vercors :
62     L15 = [ 3 , [ "Pluie" , -2 , 80 , 0 , 130 ] , [ "Pluie" , -2 , 20 , 0 , 150 ] , [ "Pluie" , -2 , 70 , 0 , 190 ] , [ "Soleil" , -2 , 30 , 1 , 210 ] , [ "Soleil" ,0 , 20 , 1 , 205 ] , [ "Soleil" ,1 , 30 , 1 , 200 ] ]
63
64     #15/02/2020 | Beaufortain :
65     L16 = [ 3 , [ "Pluie" , -1 , 100 , 0 , 140 ] , [ "Pluie" , -1 , 100 , 2 , 180 ] , [ "Pluie" , -2 , 40 , 2 , 180 ] , [ "Pluie" , -3 , 50 , 1 , 180 ] , [ "Pluie" , -2 , 40 , 1 , 245 ] , [ "Soleil" ,1 , 10 , 0 , 200 ] ]
66
67     #04/23/2022 | Beaufortain :
68     L17 = [ 1 , [ "Soleil" , -1 , 40 , 1 , 50 ] , [ "Soleil" , -1 , 30 , 1 , 50 ] , [ "Soleil" , 1 , 40 , 1 , 50 ] , [ "Soleil" ,1 , 20 , 1 , 50 ] , [ "Soleil" ,1 , 10 , 1 , 50 ] , [ "Soleil" ,1 , 40 , 1 , 50 ] ]
69
70     #24/01/2018 | Champsaur :
71     L18 = [ 3 , [ "Pluie" , -1 , 60 , 1 , 180 ] , [ "Pluie" , -1 , 70 , 2 , 180 ] , [ "Pluie" , -1 , 100 , 1 , 210 ] , [ "Pluie" ,1 , 80 , 1 , 210 ] , [ "Soleil" ,1 , 40 , 1 , 200 ] , [ "Soleil" ,1 , 20 , 0 , 200 ] ]
72
73     L.append (L13)
74     L.append (L14)
75     L.append (L15)
76     L.append (L16)
77     L.append (L17)
78     L.append (L18)
79
80     #20/12/2016 | Vanoise
81     L19 = [ 2 , [ "Soleil" ,0 , 10 , 0 , 20 ] , [ "Soleil" ,0.10 , 0 , 20 ] , [ "Soleil" ,0.30 , 1 , 20 ] , [ "Soleil" ,0.20 , 1 , 20 ] , [ "Pluie" , -4 , 50 , 0 , 20 ] , [ "Pluie" , -5 , 100 , 0 , 20 ] ]
82
83     #04/02/2020 | Maurienne
84     L20 = [ 2 , [ "Pluie" ,2 , 50 , 1 , 140 ] , [ "Pluie" ,2 , 20 , 1 , 135 ] , [ "Pluie" ,2 , 20 , 1 , 110 ] , [ "Pluie" ,2 , 50 , 1 , 20 ] , [ "Pluie" ,2 , 70 , 1 , 100 ] , [ "Pluie" , -1 , 100 , 1 , 100 ] ]
85
86     #12/01/2022 | Bauges
87     L21 = [ 2 , [ "Pluie" , -1 , 10 , 1 , 60 ] , [ "Pluie" , -1 , 20 , 1 , 70 ] , [ "Pluie" , -1 , 60 , 1 , 110 ] , [ "Soleil" , -1 , 30 , 1 , 110 ] , [ "Soleil" , -1 , 10 , 0 , 100 ] , [ "Soleil" , 1 , 10 , 1 , 100 ] ]
88
89     #03/02/2020 | Haute-Tarentaise
90     L22 = [ 4 , [ "Pluie" , -1 , 90 , 2 , 150 ] , [ "Pluie" , -1 , 140 , 2 , 165 ] , [ "Pluie" , -1 , 80 , 2 , 165 ] , [ "Pluie" , -1 , 60 , 2 , 180 ] , [ "Pluie" , -1 , 80 , 2 , 175 ] , [ "Pluie" ,1 , 90 , 2 , 175 ] ]
91
92     #05/02/2019 | Vercors
93     L23 = [ 3 , [ "Pluie" , -1 , 30 , 0 , 100 ] , [ "Pluie" , -1 , 30 , 0 , 130 ] , [ "Pluie" , -1 , 80 , 0 , 150 ] , [ "Pluie" , -1 , 20 , 1 , 190 ] , [ "Soleil" , -1 , 30 , 1 , 210 ] , [ "Soleil" ,1 , 35 , 1 , 205 ] ]
94
95     #21/12/2016 | Oisans
96     L24 = [ 2 , [ "Soleil" ,3 , 10 , 3 , 20 ] , [ "Soleil" ,3 , 10 , 3 , 20 ] , [ "Soleil" , -3 , 20 , 1 , 20 ] , [ "Soleil" ,3 , 40 , 0 , 20 ] , [ "Pluie" , -1 , 70 , 0 , 20 ] , [ "Soleil" ,1 , 20 , 0 , 20 ] ]
97
98     L.append (L19)
99     L.append (L20)
100    L.append (L21)
101    L.append (L22)
102    L.append (L23)
103    L.append (L24)
104
105
106    return L

```

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def CompareMeteo() :
5     L = DataMeteoListe()          #Liste de mes Bulletins/risques.
6     X = []                        #Liste des estimations MeteoFrance.
7     Y = []                        #Liste de mes estimations.
8     P = []                        #Liste provisoire de "Chargement".
9     Z = []                        #Liste abcisse
10    M = []                        #Liste moyenne.
11
12    for k in range ( len (L) ) :      #Boucle utile au remplissage de la liste d'estimation MeteoFrance.
13        X.append( L[k][0] )
14
15    for k in range ( len (L) ) :      #Boucle utile au remplissage de la liste de mes estimations.
16        for i in range (1, len(L[0]) ) :
17            P.append( L[k][i] )           #Cette liste devient pour chaque liste "datas" la liste contenant le bulletin météo.
18            Y.append( IndiceRisqueAvalanche(P) )
19            P = []
20
21    for k in range (len(L)) :        #Liste d'abcisse pour l'essai n°1 jusqu'à len(L).
22        Z.append(k+1)
23
24    #Liste de moyennes :
25    for k in range ( len(L) ) :
26        M.append( (X[k] + Y[k]) / 2 )
27    #Calcul moyenne écart :
28    mo = 0
29    for k in range ( len(L) ) :
30        mo += (X[k] - Y[k])
31
32
33
34
35 #Traçage
36
37    x = np.array(X)
38    y = np.array(Y)
39    z = np.array(Z)
40    m = np.array(M)
41
42    plt.plot( z, x , "g:o" , label = "estimations MeteoFrance" )
43    plt.plot( z, y , "b:o" , label = "Mes estimations" )
44    plt.plot( z, m , "k--" , label = "Moyenne", linewidth=0.25 )
45
46
47    plt.ylabel( "Indice Risque Avalanche" )
48    plt.xlabel( "Essai n°" )
49
50    plt.ylim(-1, 6)
51
52
53    plt.legend()
54
55    plt.title( "Comparaison résultats/réalité" )
56
57
58    plt.show()
59
60    return (mo / len(L))
```

```
1 from math import atan, pi
2 def DeterminePentes(M) :
3     PH = []      #Pentes horizontales, les deux listes seront des listes de 2-uplets de la forme [orientation(est/ouest, pente(en degrés)).
4     PV = []      #Pentes verticales.
5     d = 5        #Distance en mètres entre 2 points.
6     o = "o"       #Orientation provisoire.
7     p = 0         #Pente provisoire.
8     m = 50
9
10    for k in range (len (M)) : #pour chaque ligne
11        PH.append( [] )
12        for i in range (len(M[0])) -1 : #Pour chaque colonnes jusqu'à l'avant dernière.
13            if M[k][i] < M[k][i+1] :
14                o = "Ouest"
15                p = ( atan( ( ( M[k][i+1] - M[k][i] ) / m ) ) * 180 /  pi
16                PH[k].append( [o,p] )
17            else :
18                o = "Est"
19                p = ( atan( ( ( M[k][i] - M[k][i+1] ) / m ) ) * 180 /  pi
20                PH[k].append( [o,p] )
21
22    for k in range ( len(M) - 1 ) :
23        PV.append( [] )
24        for i in range ( len (M[0]) ) :
25            if M[k][i] < M[k+1][i] :
26                o = "Nord"
27                p = ( atan( ( ( M[k+1][i] - M[k][i] ) / m ) ) * 180 /  pi
28                PV[k].append( [o,p] )
29            else :
30                o = "Sud"
31                p = ( atan( ( ( M[k][i] - M[k+1][i] ) / m ) ) * 180 /  pi
32                PV[k].append( [o,p] )
33
34    return ( [PH,PV] )
35
```

```

1 #M la matrice d'élévation et B le bulletin météo
2 def PentesRisques (M,B) :
3     PH = DeterminePentes(M)[0]
4     PV = DeterminePentes(M)[1]
5     I = IndiceRisqueAvalanche (B)
6
7     IV = False           #Nous prenons le vent en compte seulement si il est à plus de 40 km/h-1
8     if B[5][2] >= 40 :
9         IV = True
10
11    #Sens du vent
12
13    SV = B[5][3]
14    DV = 0                #Danger du vent = Opposé à sa direction
15    if SV == [0] :
16        DV = "Sud"
17    if SV == [1] :
18        DV = "Nord"
19    if SV == [2] :
20        DV = "Ouest"
21    if SV == [3] :
22        DV = "Est"
23
24    #Générions deux grille de 0, de la taille de la matrice de pente
25    PH0 = []
26    for k in range (len(PH)) :
27        PH0.append( [] )
28        for u in range ( len(PH[0]) ) :
29            PH0[k].append( 0 )
30
31    PV0 = []
32    for k in range (len(PV)) :
33        PV0.append( [] )
34        for u in range ( len(PV[0]) ) :
35            PV0[k].append( 0 )
36
37    for i in range (len(PH)) :
38        for j in range (len(PH[0])) :
39
40            if I == 1 :
41                if PH[i][j][1] > 35 and IV and PH[i][j][0] == DV :
42                    PH0[i][j] = 1
43                if PH[i][j][1] > 40 :
44                    PH0[i][j] = 1
45
46
47            if I == 2 :
48                if PH[i][j][1] > 30 and IV and PH[i][j][0] == DV :
49                    PH0[i][j] = 1
50                if PH[i][j][1] > 35 :
51                    PH0[i][j] = 1
52
53                if PH[i][j][1] > 35 and IV and PH[i][j][0] == DV :
54                    PH0[i][j] = 2
55                if PH[i][j][1] > 40 :
56                    PH0[i][j] = 2
57
58
59            if I == 3 :
60                if PH[i][j][1] > 25 and IV and PH[i][j][0] == DV :
61                    PH0[i][j] = 1
62                if PH[i][j][1] > 30 :
63                    PH0[i][j] = 1
64
65                if PH[i][j][1] > 30 and IV and PH[i][j][0] == DV :
66                    PH0[i][j] = 2
67                if PH[i][j][1] > 35 :
68                    PH0[i][j] = 2
69
70            if I == 4 :
71                if PH[i][j][1] > 20 and IV and PH[i][j][0] == DV :
72                    PH0[i][j] = 1
73                if PH[i][j][1] > 25 :
74                    PH0[i][j] = 1
75
76                if PH[i][j][1] > 25 and IV and PH[i][j][0] == DV :
77                    PH0[i][j] = 2
78                if PH[i][j][1] > 30 :
79                    PH0[i][j] = 2
80
81
82
83    for i in range (len(PV) ) :
84        for j in range (len(PV[0])) :
85            if I == 1 :
86                if PV[i][j][1] > 30 and IV and PV[i][j][0] == DV and PV[i][j][0] == "Nord" :
87                    PV0[i][j] = 1
88                if PV[i][j][1] > 35 and PV[i][j][0] == "Nord" :
89                    PV0[i][j] = 1
90                if PV[i][j][1] > 35 and IV and PV[i][j][0] == DV :
91                    PV0[i][j] = 1
92                if PV[i][j][1] > 40 :
93                    PV0[i][j] = 1
94
95
96            if I == 2 :
97                if PV[i][j][1] > 25 and I and PV[i][j][0] == DV and PV[i][j][0] == "Nord" :
98                    PV0[i][j] = 1
99                if PV[i][j][1] > 30 and PV[i][j][0] == "Nord" :
100                   PV0[i][j] = 1
101                if PV[i][j][1] > 30 and I and PV[i][j][0] == DV :
102                    PV0[i][j] = 1
103                if PV[i][j][1] > 35 :
104                    PV0[i][j] = 1
105
106                if PV[i][j][1] > 30 and IV and PV[i][j][0] == DV and PV[i][j][0] == "Nord" :
107                    PV0[i][j] = 2
108                if PV[i][j][1] > 35 and PV[i][j][0] == "Nord" :
109                    PV0[i][j] = 2
110                if PV[i][j][1] > 35 and IV and PV[i][j][0] == DV :
111                    PV0[i][j] = 2
112                if PV[i][j][1] > 40 :
113                    PV0[i][j] = 2
114
115
116            if I == 3 :
117                if PV[i][j][1] > 20 and IV and PV[i][j][0] == DV and PV[i][j][0] == "Nord" :
118                    PV0[i][j] = 1
119                if PV[i][j][1] > 25 and PV[i][j][0] == "Nord" :
120                    PV0[i][j] = 1
121                if PV[i][j][1] > 25 and IV and PV[i][j][0] == DV :
122                    PV0[i][j] = 1
123                if PV[i][j][1] > 30 :
124                    PV0[i][j] = 1
125
126                if PV[i][j][1] > 25 and IV and PV[i][j][0] == DV and PV[i][j][0] == "Nord" :
127                    PV0[i][j] = 2
128                if PV[i][j][1] > 30 and PV[i][j][0] == "Nord" :
129                    PV0[i][j] = 2
130                if PV[i][j][1] > 30 and IV and PV[i][j][0] == DV :
131                    PV0[i][j] = 2
132                if PV[i][j][1] > 35 :
133                    PV0[i][j] = 2
134
135
136            if I == 4 :
137                if PV[i][j][1] > 15 and IV and PV[i][j][0] == DV and PV[i][j][0] == "Nord" :
138                    PV0[i][j] = 1
139                if PV[i][j][1] > 20 and PV[i][j][0] == "Nord" :
140                    PV0[i][j] = 1
141                if PV[i][j][1] > 20 and IV and PV[i][j][0] == DV :
142                    PV0[i][j] = 1
143                if PV[i][j][1] > 25 :
144                    PV0[i][j] = 1
145
146                if PV[i][j][1] > 20 and IV and PV[i][j][0] == DV and PV[i][j][0] == "Nord" :
147                    PV0[i][j] = 2
148                if PV[i][j][1] > 25 and PV[i][j][0] == "Nord" :
149                    PV0[i][j] = 2
150                if PV[i][j][1] > 25 and IV and PV[i][j][0] == DV :
151                    PV0[i][j] = 2
152                if PV[i][j][1] > 30 :
153                    PV0[i][j] = 2
154
155
156    return ([B,I,PH0,PV0])
157
158
159
160

```