



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
MATEMATINĖS STATISTIKOS KATEDRA

Julija Janeiko

DAUGIAMAČIŲ KRYPTINIŲ DUOMENŲ MODELIAVIMAS

Multidimensional directional data modeling

Baigiamasis magistro darbas

Taikomosios statistikos studijų programa, valstybinis kodas 6211AX009

Duomenų mokslo specializacija

Statistikos studijų kryptis

Vilnius, 2021

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
MATEMATINĖS STATISTIKOS KATEDRA

TVIRTINU

Katedros vedėjas

(Parašas)

doc. dr. Kazimieras Padvelskis
(Vardas, pavardė)

(Data)

Julija Janeiko

DAUGIAMAČIŲ KRYPTINIŲ DUOMENŲ
MODELIAVIMAS

Multidimensional directional data modeling

Baigiamasis magistro darbas

Taikomosios statistikos studijų programa, valstybinis kodas 6211AX009

Duomenų mokslo specializacija

Statistikos studijų kryptis

Vadovas

doc. dr. Tomas Rekašius
(Pedagoginis vardas, vardas, pavardė)

(Parašas)

(Data)

Lietuvių kalbos konsultantė

dr. Vaida Buivydienė
(Pedagoginis vardas, vardas, pavardė)

(Parašas)

(Data)

Vilnius, 2021

Vilniaus Gedimino technikos universitetas
Fundamentinių mokslų fakultetas
Matematinės statistikos katedra

ISBN ISSN
Egz. sk.
Data-....-....

Antrosios pakopos studijų **Taikomosios statistikos** programos magistro baigiamasis darbas
Pavadinimas **Daugiamačių kryptinių duomenų modeliavimas**
Autorius **Julija Janeiko**
Vadovas **doc. dr. Tomas Rekašius**

Kalba
☒ lietuvių
☐ užsienio

Anotacija

Baigiamajame magistro darbe yra nagrinėjami daugiamačių kryptinių duomenų modeliavimo metodai. Cilindriniais ir sferiniams duomenims yra pritaikomi k-vidurkių bei EM klasterizavimo metodų kryptiniai analogai. Kryptiniai klasterizavimo metodai yra lyginami tarpusavyje bei su klasikiniu tiesiniu klasterizavimo algoritmu. Taip pat, cilindrinį bei sferinį duomenų klasifikavimui yra pritaikomi kryptiniai diskriminantinės analizės ir logistinės regresijos metodų analogai. Jie yra lyginami tarpusavyje ir su klasikiniu tiesiniu klasifikavimo algoritmu. Be to, darbe cilindriniais duomenims yra pritaikomas kryptinių sprendimų medžių algoritmas. Analizei yra naudojami sugeneruoti sferiniai bei cilindriniai duomenys, taip pat ir realūs duomenų rinkiniai: žvaigždžių duomenys ir meteorologiniai duomenys. Skaičiavimai atlikti naudojant programavimo kalbą R, versija 3.6.3.

Darbą sudaro 6 dalys: įvadas, teorinė dalis, praktinė dalis, išvados, literatūros sąrašas.

Darbo apimtis – 108 p. teksto be priedų, 46 lentelės, 60 iliustracijų, 23 bibliografiniai šaltiniai.

Atskirai pridedami darbo priedai.

Raktiniai žodžiai: kryptinė statistika, cilindriniai duomenys, sferiniai duomenys, kryptinis klasifikavimas, kryptinis klasterizavimas, kryptiniai sprendimų medžiai.

Turinys

Iliustracijų sąrašas	5
Lentelių sąrašas	8
Įvadas	11
1. Teorinė dalis	13
1.1. Duomenų tipai ir struktūros	13
1.2. Tikimybių teorija kryptiniams dydžiams	17
1.3. Aprašomoji statistika	19
1.3.1. Išplėtimas daugiamačiams kryptiniams dydžiams	20
1.4. Pagrindiniai kryptiniai pasiskirstymai	21
1.4.1. Vienmačiai pasiskirstymai	21
1.4.2. Daugiamačiai pasiskirstymai	23
1.5. Klasterizavimas kryptiniams duomenims	28
1.5.1. EM klasterizavimas	28
1.5.2. k-vidurkių klasterizavimo metodas	32
1.6. Kryptinių duomenų klasifikavimas	34
1.6.1. Diskriminantinė analizė	34
1.6.2. Logistinė regresija	37
1.7. Kryptinių duomenų sprendimų medžiai	38
2. Praktinė dalis	41
2.1. Duomenys	41
2.1.1. Sumodeliuoti duomenys	41
2.1.2. Realus duomenys	43
2.2. Sferinių duomenų klasterizavimas	51
2.2.1. EM klasterizavimas	51
2.2.2. Klasterizavimas naudojant k-vidurkių metodą	52
2.2.3. Palyginimas su klasikiniu klasterizavimu	54
2.2.4. Metodų tarpusavyje palyginimas	55
2.3. Sferinių duomenų klasifikavimas	56
2.3.1. Diskriminantinė analizė	56
2.3.2. Logistinė regresija	59
2.3.3. Palyginimas su klasikiniu klasifikavimu	62
2.3.4. Metodų tarpusavyje palyginimas	64
2.4. Cilindrinų duomenų klasterizavimas	65

2.4.1.	EM klasterizavimas	65
2.4.2.	Klasterizavimas naudojant k-vidurkių metodą	68
2.4.3.	Palyginimas su klasikiniu klasterizavimu	71
2.4.4.	Metodų tarpusavyje palyginimas	72
2.5.	Cilindrinų duomenų klasifikavimas	73
2.5.1.	Diskriminantinė analizė	73
2.5.2.	Logistinė regresija	85
2.5.3.	Palyginimas su klasikiniu klasifikavimu	89
2.5.4.	Metodų tarpusavyje palyginimas	93
2.6.	Kryptiniai sprendimų medžiai	94
Išvados		101
Literatūra		102
A. Programinis kodas		104

Iliustracijų sąrašas

1.	Kryptinis stebinyš ant apskritimo	14
2.	Daugiamatis kryptinis stebinyš ant cilindro	15
3.	Daugiamatis kryptinis stebinyš ant toro	16
4.	Daugiamatis kryptinis stebinyš ant sferos	16
5.	Kryptinio tolygiojo pasiskirstymo tankis	21
6.	Susuktas normalusis pasiskirstymas	22
7.	Von Mises pasiskirstymas	23
8.	Von Mises-Fisher ant sferos	24
9.	Von Mises-Normalusis tankis	25
10.	Dalinai susuktas normalusis (SWG) tankis	27
11.	Sprendimų medžio modelio schema	39
12.	Sugeneruoti sferiniai duomenys su 3 grupėmis	42
13.	Sugeneruoti cilindriniai duomenys su 3 grupėmis	43
14.	Sudėtinių žvaigždžių suskirstytų pagal jų kodus padėtis dangaus sferoje . .	44
15.	Meteorologinių duomenų grupavimo pagal metų laikus pasiskirstymai ant cilindro	46
16.	Meteorologinių duomenų grupavimo pagal metų laikus pasiskirstymai . . .	47
17.	Meteorologinių duomenų grupavimas pagal metų laikus ant sumažintos dimensijos cilindrinio duomenų	48
18.	Dieninių meteorologinių duomenų grupavimo pagal metų laikus pasiskirstymai ant cilindro	49
19.	Dieninių meteorologinių duomenų grupavimo pagal metų laikus pasiskirstymai	50
20.	Dieninių meteorologinių duomenų grupavimas pagal metų laikus ant sumažintos dimensijos cilindrinio duomenų	51
21.	EM klasterizavimas pritaikytas von Mises-Fisher pasiskirstymo duomenims	52
22.	Sferinis k-vidurkių klasterizavimas	53
23.	Tiesinis k-vidurkių klasterizavimas	54
24.	Klasterizavimo ant sferos metodų charakteristikų tankiai	55
25.	Sferinės tiesinės diskriminantinės analizės klasifikavimas ant sumodeliuotų duomenų	56
26.	Žvaigždžių klasifikavimas naudojant sferinę tiesinę diskriminantinę analizę	58
27.	Sferinės logistinės regresijos modelio rezultatas sumodeliuotiems duomenims	59
28.	Žvaigždžių klasifikavimas naudojant sferinę logistinę regresiją	60
29.	Žvaigždžių klasifikavimas naudojant sferinę logistinę regresiją su papildomais kintamaisiais	62

30.	Žvaigždžių klasifikavimas naudojant klasikinę logistinę regresiją su papildomais kintamaisiais	63
31.	Klasifikavimo ant sferos metodų charakteristikų tankiai	64
32.	Cilindrinų duomenų klasterizavimas naudojant EMvMN metodą	66
33.	Cilindrinų duomenų klasterizavimas naudojant EM-SWG metodą	67
34.	Cilindrinų duomenų klasterizavimas naudojant fiksuotą cyk-vidurkių metodą	69
35.	Cilindrinų duomenų klasterizavimas naudojant cyk-vidurkių metodą	70
36.	Cilindrinų duomenų klasterizavimas naudojant klasikinį k-vidurkių metodą	71
37.	Klasterizavimo ant cilindro metodų charakteristikų tankiai	72
38.	Klasifikavimas naudojant cilindrinę diskriminantinę von Mises-Normaliąją analizę ant sugeneruotų duomenų	74
39.	Meteorologinių duomenų klasifikavimas naudojant cilindrinę diskriminantinę von Mises-Normaliąją analizę	75
40.	Meteorologinių duomenų klasifikavimas naudojant hipercilindrinę diskriminantinę von Mises-Normaliąją analizę	77
41.	Dieninių meteorologinių duomenų klasifikavimas naudojant cilindrinę diskriminantinę von Mises-Normaliąją analizę	78
42.	Dieninių meteorologinių duomenų klasifikavimas naudojant hipercilindrinę diskriminantinę von Mises-Normaliąją analizę	80
43.	Klasifikavimas naudojant cilindrinę diskriminantinę SWG analizę ant sugeneruotų duomenų	81
44.	Meteorologinių duomenų klasifikavimas naudojant cilindrinę diskriminantinę SWG	82
45.	Dieninių meteorologinių duomenų klasifikavimas naudojant cilindrinę diskriminantinę SWG	84
46.	Klasifikavimas naudojant cilindrinę logistinę regresiją ant sugeneruotų duomenų	85
47.	Meteorologinių duomenų klasifikavimas naudojant cilindrinę logistinę regresiją	87
48.	Dieninių meteorologinių duomenų klasifikavimas naudojant cilindrinę logistinę regresiją	88
49.	Meteorologinių duomenų klasifikavimas naudojant klasikinę logistinę regresiją	90
50.	Dieninių meteorologinių duomenų klasifikavimas naudojant klasikinę logistinę regresiją	92
51.	Klasifikavimo ant cilindro metodų charakteristikų tankiai	93
52.	Kintamojo θ bendras pasiskirstymas ir DT von Mises pasiskirstymas	95
53.	Papildomai kintamųjų x_1 ir x_2 , pagal kuriuos įmanoma atskirti kintamojo θ grupes, sklaidos diagrama	95
54.	Medžių modelis su sumodeliuotais duomenimis	96

55.	Tikrų atsako kintamojo θ grupių pasiskirstymai	97
56.	Suprognuotų atsako kintamojo θ grupių pasiskirstymai	97
57.	Vėjo krypties pasiskirstymas ir DT von Mises pasiskirstymas	98
58.	Vėjo krypties medžių modelis	99
59.	Suprognuotų vėjo krypčių grupių pasiskirstymai dieniniams duomenims .	99
60.	Suprognuotų vėjo krypčių grupių pasiskirstymai	100

Lentelių sąrašas

1.	Sugeneruotų sferinių duomenų pasiskirstymų parametrai	41
2.	Sugeneruotų cilindrinų duomenų pasiskirstymų parametrai	42
3.	Sudėtinių žvaigždžių apjungti grupavimo kodai ir jų dažniai	44
4.	Meteorologinių duomenų metų laikai ir jų dažniai	45
5.	Pirmos pagrindinės komponentės iš meteorologinių tiesinių kintamųjų koeficientai	47
6.	Dieninių meteorologinių duomenų metų laikai ir jų dažniai	48
7.	Pirmos pagrindinės komponentės iš meteorologinių tiesinių kintamųjų koeficientai	50
8.	Sferinio EM klasterizavimo rezultatai	52
9.	Sferinio k-vidurkių metodo klasterizavimo rezultatai	53
10.	Klasikinio k-vidurkių metodo klasterizavimo rezultatai	54
11.	Klasterizavimo ant sferos metodų vidutinės charakteristikos	55
12.	Sumodeliuotų duomenų sferinės diskriminantinės analizės tikslumo matai .	57
13.	Žvaigždžių sferinės diskriminantinės analizės klasifikavimo tikslumo matai .	58
14.	Sumodeliuotų duomenų sferinės logistinės regresijos klasifikavimo tikslumo matai	59
15.	Žvaigždžių klasifikavimo su sferine logistine regresija klasifikavimo tikslumo matai	60
16.	Logistinės regresijos modelio su papildomais kintamaisiais parametru reikšmingumo hipotezės p-reikšmės	61
17.	Žvaigždžių klasifikavimo naudojant sferinę logistinę regresiją su papildomais kintamaisiais klasifikavimo tikslumo matai	62
18.	Žvaigždžių klasifikavimo naudojant klasikinę logistinės regresijos modelį su papildomais kintamaisiais klasifikavimo tikslumo matai	63
19.	Klasifikavimo ant sferos metodų vidutinės charakteristikos	65
20.	Suklasterizuotų duomenų von Mises-Normaliojo pasiskirstymo parametrai kiekvienai grupei	66
21.	Suklasterizuotų duomenų SWG pasiskirstymo vidurkių parametrai kiekvienai grupei	68
22.	Fiksuoto cik-vidurkių metodo klasterizavimo rezultatai: klasterių dydžiai ir centrai	69
23.	cik-vidurkių metodo klasterizavimo rezultatai: klasterių dydžiai ir centrai .	70
24.	Klasikinio k-vidurkių metodo klasterizavimo rezultatai: klasterių dydžiai ir centrai	71
25.	Klasterizavimo ant cilindro metodų vidutinės charakteristikos	73

26.	Diskriminantinės von Mises-Normaliosios analizės ant sugeneruotų duomenų klasifikavimo tikslumo matai	74
27.	Meteorologinių duomenų naudojant cilindrinę diskriminantinę von Mises-Normaliąją analizę klasifikavimo tikslumo matai	76
28.	hipercilindrinės diskriminantinės analizės ant meteorologinių duomenų parametrų reikšmingumo hipotezės minimalios p-reikšmės	76
29.	Meteorologinių duomenų naudojant hipercilindrinę diskriminantinę von Mises-Normaliąją analizę klasifikavimo tikslumo matai	77
30.	Dieninių meteorologinių duomenų naudojant cilindrinę diskriminantinę von Mises-Normaliąją analizę klasifikavimo tikslumo matai	78
31.	hipercilindrinės diskriminantinės analizės ant dieninių meteorologinių duomenų parametrų reikšmingumo hipotezės minimalios p-reikšmės	79
32.	Dieninių meteorologinių duomenų naudojant hipercilindrinę diskriminantinę von Mises-Normaliąją analizę klasifikavimo tikslumo matai	80
33.	Diskriminantinės SWG analizės ant sugeneruotų duomenų klasifikavimo tikslumo matai	82
34.	Meteorologinių duomenų naudojant cilindrinę diskriminantinę SWG klasifikavimo tikslumo matai	83
35.	Dieninių meteorologinių duomenų naudojant cilindrinę diskriminantinę SWG klasifikavimo tikslumo matai	84
36.	Cilindrinė logistinė regresija ant sugeneruotų duomenų klasifikavimo tikslumo matai	86
37.	Cilindrinės logistinės regresijos ant meteorologinių duomenų parametrų reikšmingumo hipotezės p-reikšmės	86
38.	Meteorologinių duomenų naudojant cilindrinę logistinę regresiją klasifikavimo tikslumo matai	87
39.	Cilindrinės logistinės regresijos ant dieninių meteorologinių duomenų parametrų reikšmingumo hipotezės p-reikšmės	88
40.	Dieninių meteorologinių duomenų naudojant cilindrinę logistinę regresiją klasifikavimo tikslumo matai	89
41.	Klasikinės logistinės regresijos ant meteorologinių duomenų parametrų reikšmingumo hipotezės p-reikšmės	90
42.	Meteorologinių duomenų naudojant klasikinę logistinę regresiją klasifikavimo tikslumo matai	91
43.	Klasikinės logistinės regresijos ant dieninių meteorologinių duomenų parametrų reikšmingumo hipotezės p-reikšmės	91
44.	Dieninių meteorologinių duomenų naudojant klasikinę logistinę regresiją klasifikavimo tikslumo matai	92
45.	Klasifikavimo ant cilindro metodų vidutinės charakteristikos	94

46.	Atsako kintamojo θ prognozuotų grupių pasiskirstymo parametrai	96
-----	---	----

Išvadas

Kryptinė statistika – tai statistikos sritis, nagrinėjanti duomenis, kurie gali būti reprezentuoti vienetiniais vektoriais erdvėje $\mathbb{R}^n$. Vienas iš paprasčiausių atvejų yra, kai duomenis galima pavaizduoti ant vienetinio apskritimo. Tokie duomenys gali būti gaunami skirtingais metodais, tačiau du pagrindiniai yra darant matavimus su tokiais įrankiais kaip kompasas ir laikrodis. Dar vienas populiarus duomenų tipas – tai sferiniai duomenys, kurie yra gaunami stebint kokį nors reiškinį ant Žemės paviršiaus arba matuojant kryptis erdvėje. Tokie duomenys yra vadinami *kryptiniais*.

Kryptiniai duomenys turi daug naujų ir unikalių bruožų, kurie nėra būdingi tiesiniams duomenims. Jie gali būti reprezentuojami ant vienetinio apskritimo, dėl to kampų skaitinės reikšmės gali būti nevienareikšmės. Jos priklauso nuo pasirinktos nulinės krypties ir sukimosi krypties, t. y. ar kampai buvo matuojami pagal laikrodžio rodyklę, ar prieš. Taip pat verta paminėti, kad kryptiniai duomenys yra periodiški bei jų pradžios ir galo taškai sutampa. Tokie duomenų išskirtiniai bruožai reikalauja naujų statistinės analizės ir modeliavimo metodų, nes įprasti standartinės tiesinės statistikos metodai dažnai duos klaidingus arba visai beprasmius rezultatus.

Kryptiniai duomenys, kaip ir *tiesiniai*, gali būti vienmačiai ir daugiamačiai. Vienmačiai duomenys yra išsidėstę ant apskritimo, o daugiamačių populiariausios duomenų struktūros yra: cilindriniai, toroidiniai arba sferiniai duomenys.

Kryptinė statistika yra taikoma daugelyje sričių. Tarp jų yra astronomija, geologija, medicinos mokslai, meteorologija, okeanografija, geografija, biologija, ekologija, fizika ir proteomika.

Darbo aktualumas. Kryptinės statistikos analizės taikymai yra svarbūs, nes įprastos tiesinės statistikos metodai tinka ne visiems duomenų tipams. Kryptinė statistika nagrinėja vieną tokį tipą – periodinius duomenis, kuriems tiesinė statistika negali duoti gerų rezultatų.

Darbo tikslas. Pritaikyti skirtingus modeliavimo algoritmus daugiamačiams kryptiniams duomenims bei palyginti jų veikimą su klasikineis metodais.

Uždaviniai.

1. Pritaikyti sferinius EM bei k-vidurkių klasterizavimo metodų analogus ir palyginti juos su klasikiniu klasterizavimo algoritmu.
2. Pritaikyti cilindrinčius EM bei k-vidurkių klasterizavimo metodų analogus ir palyginti juos su klasikiniu klasterizavimo algoritmu.

3. Pritaikyti diskriminantinės analizės bei logistinės regresijos sferinius analogus ir palyginti juos su klasikiniu klasifikavimo algoritmu.
4. Pritaikyti diskriminantinės analizės bei logistinės regresijos cilindrinis analogus ir palyginti juos su klasikiniu klasifikavimo algoritmu.
5. Pritaikyti kryptinio regresijos medžio metodą skrituliniam kintamajam modeliuoti nuo tiesinių prediktorių.

1. Teorinė dalis

Kryptiniai (angl. *circular*) duomenys turi daug naujų ir unikalių bruožų, kurie nėra būdingi tiesiniams duomenims. Kaip jau buvo minėta įvade, kryptiniai duomenys gali būti reprezentuojami ant vienetinio apskritimo, dėl to kampų skaitinės reikšmės gali būti nevienareikšmės. Jos priklauso nuo pasirinktos nulinės krypties ir sukimosi krypties, t. y. ar kampai buvo matuojami pagal laikrodžio rodyklę, ar prieš. Taip pat verta paminėti, kad kryptiniai duomenys yra periodiškai bei jų pradžios ir galo taškai sutampa. Daugiamačiams kryptiniams duomenims: cilindriniams, toroidiniams bei sferiniams, taip pat yra būdingi panašūs bruožai. Tokie duomenų išskirtiniai bruožai reikalauja naujų statistinės analizės ir modeliavimo metodų, nes įprasti standartinės tiesinės statistikos metodai dažnai duos klaidingus arba visai beprasmius rezultatus. Šitoje darbo dalyje bus aprašyti klasikiniai modeliavimo metodai bei jų analogai, tinkantys kryptiniams duomenims.

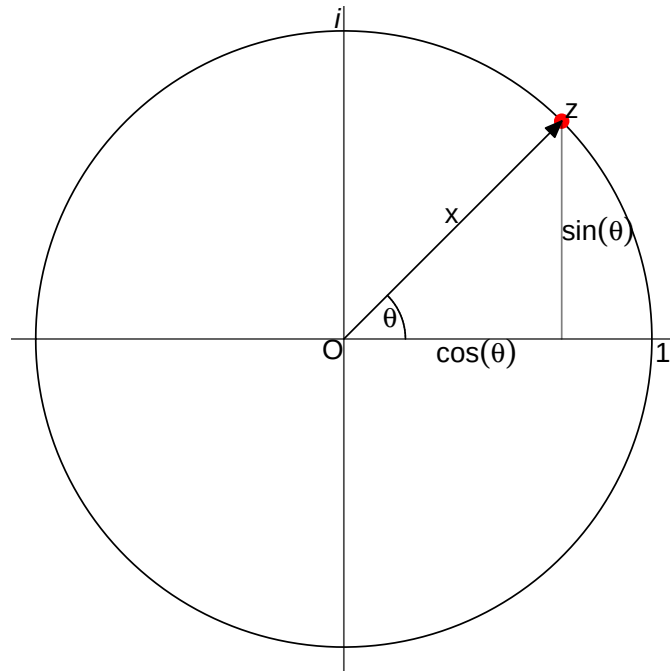
1.1. Duomenų tipai ir struktūros

Kaip jau buvo minėta įvade, kryptiniai duomenys gali būti vienmačiai ir daugiamačiai. Jie taip gali turėti skirtingas struktūras. Toliau yra aprašomos kelios populiarios kryptinių duomenų struktūros.

Skrituliniai duomenys. Duomenys, kuriuos galima reprezentuoti ant vienetinio apskritimo yra vadinami skrituliniais duomenimis (*angl. circular data*). Tokio tipo duomenų pavyzdžiai yra:

- kryptys išmatuotos su kompasu – vėjo kryptys, migruojančių paukščių skrydžio kryptys;
- įvairūs kampai pateikti laipsniais ar radianais nuo pasirinktos pradžios – dalinės atomų masės;
- laiko tarpai – maksimalios temperatūros laikas arba uraganų atsiradimas per metus.

Skrituliniai duomenys yra vienmačiai. Juos yra patogiau apibrėžti kampais θ_j , kurie įgyja reikšmes intervale $[0, 2\pi)$ (arba intervale $[-\pi, \pi)$). Intervalo pradžia ir atitinkamai pabaiga yra pasirenkami ir nedaro įtakos analizės rezultatams. Skrituliniai duomenys taip pat gali būti apibrėžiami vektoriais $\mathbf{x}_j = (\cos \theta_j, \sin \theta_j)^T$ arba kompleksiniais skaičiais $z_j = e^{i\theta_j}$. 1 pav. yra pavaizduotas vienas toks taškas.



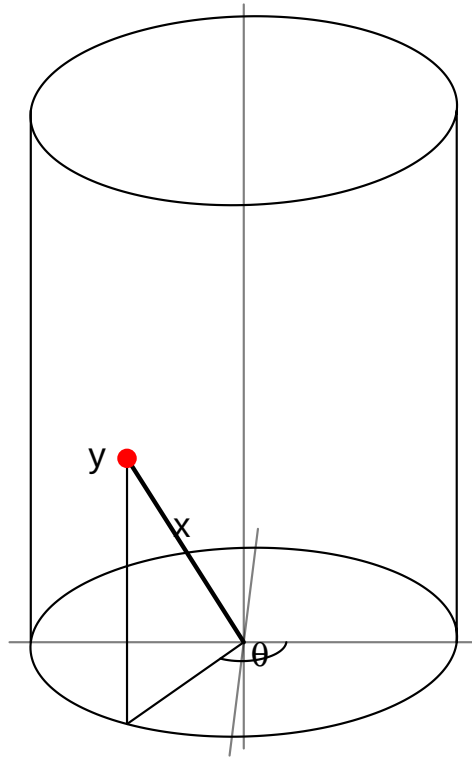
1 pav. Kryptinis stebiny ant apskritimo

Cilindriniai duomenys. Duomenys, kuriuos galima reprezentuoti ant cilindro yra vadinami cilindriniais duomenimis (*angl. cylindrical data*). Tokio tipo duomenys yra dvimačiai. Juos sudaro vienas tiesinis kintamasis y ir vienas skritulinis θ . Cilindrinis duomenis galima išreikšti vektoriumi $\mathbb{R}^3$ erdvėje: $\mathbf{x} = (\cos \theta, \sin \theta, y)^T$.

Cilindrinų duomenų pavyzdžiai yra:

- vektorių ilgiai ir kryptys – vėjo greitis ir kryptis.
- laiko eilutės, kurių laiko intervalai nėra vienodi – kritulių poros laikas ir intensyvumas.

2 pav. yra pavaizduota, kaip atrodo vienas cilindrinų duomenų taškas.



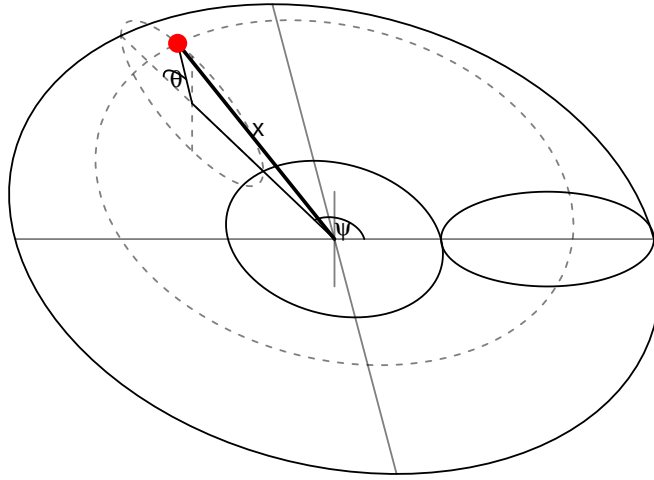
2 pav. Daugiamatis kryptinis stebiny ant cilindro

Toroidiniai duomenys. Duomenys, kuriuos galima reprezentuoti ant toro yra vadinami toroidiniais duomenimis (*angl. toroidal data*). Tokio tipo duomenys irgi yra dvimačiai. Juos sudaro du skrituliniai kintamieji: θ ir ψ .

Toroidinių duomenų pavyzdžiai yra:

- baltymo ir ligando jungimosi kampai;
- vėjo ir bangų kryptys.

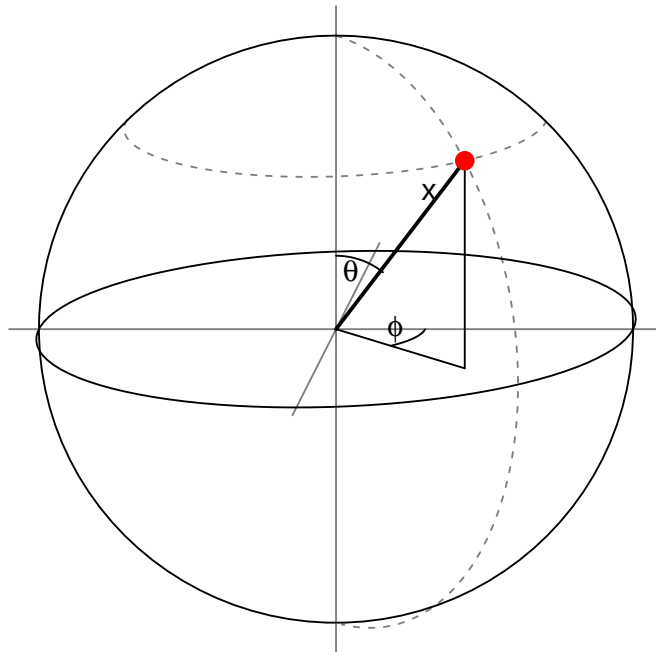
3 pav. yra pavaizduota, kaip atrodo vienas toroidinis duomenų taškas.



3 pav. Daugiamatis kryptinis stebiny ant toro

Bendru atveju, galima nagrinėti ir daugiau nei du skritulinius kintamuosius kartu. Tokiu atveju duomenys bus hipertoroidiniai.

Sferiniai duomenys. Duomenys, kuriuos galima reprezentuoti ant sferos yra vadinami sferiniais duomenimis (*angl. spherical data*). Tokio tipo duomenys taip pat yra dvimačiai. Juos sudaro vienas skritulinis kintamasis $\phi \in [0, 2\pi]$ ir vienas ašinis (*angl. axial*) kintamasis $\theta \in [0, \pi]$, kuris yra išsidėstęs ant pusės apskritimo. Sferines koordinates galima išreikšti vektoriumi $\mathbb{R}^3$ erdvėje: $\mathbf{x} = (\sin \theta \cos \phi, \sin \theta \sin \phi, \cos \theta)^T$. Kaip atrodo vienas sferinių duomenų taškas, yra pavaizduota 4 pav.



4 pav. Daugiamatis kryptinis stebiny ant sferos

Galima pastebėti, kad geografinėje terminologijoje, kampas θ žymi ko-platumą (taip, kad $\frac{\pi}{2} - \theta$ žymi platumą), o kampas ϕ – ilgumą. Taip pat, taškas su koordinatėmis $(-\theta, \phi)$ yra ekvivalentus taškui $(\theta, \phi + \pi)$.

Sferinių duomenų pavyzdžiai yra:

- reiškiniai stebimi ant Žemės paviršiaus – paleomagnetizmo kryptys uolose.
- įvairios kryptys išdėstytos erdvėje – astronominių objektų padėtys dangaus sferoje.

Bendru atveju, duomenys gali turėti ir daugiau dimensijų hipersferinėje erdvėje.

Šiame darbe toliau yra nagrinėjami tik du daugiamačių kryptinių duomenų tipai: sferiniai ir cilindriniai dydžiai.

1.2. Tikimybių teorija kryptiniams dydžiams

Šitame skyriuje yra apibrėžiamos pagrindinės tikimybinių pasiskirstymų sąvokos kryptiniams atsitiktiniams dydžiams. Ši dalis yra parašyta remiantys kelių statistikų darbais: Mardia (1972); Jammalamadaka & Sengupta (2001).

Pirmiausiai apibrėžiame kas yra atsitiktinis kampas ir funkcijos, kurios nusako šio dydžio pasiskirstymą ant vienetinio apskritimo.

Atsitiktinis kampas. Tegul ξ yra atsitiktinis dydis ant tiesės. Tada dydis $\Theta = \xi \bmod 2\pi$ yra vadinamas atsitiktiniu kampu. Ant vienetinio apskritimo šis kampas atitinka kompleksinį skaičių $Z = e^{i\Theta} = \cos \Theta + i \sin \Theta$ arba vektorių $X = (\cos \Theta, \sin \Theta)^T$. Paprastumo dėlei, kampai yra reprezentuojami „matematiškai“, t. y. nulinė kryptis atitinka teigiamą horizontalią ašį ir kampai yra išmatuoti prieš laikrodžio rodyklę radianais.

Toliau laikysime, kad atsitiktinis kampas yra absoliučiai tolydus ir nagrinėsime tik šį atvejį.

Kryptinė pasiskirstymo funkcija F . Funkcija F apibrėžiama panašiai kaip ir tiesiniu atveju, tačiau ji turi kelias naujas savybes būdingas kryptiniams dydžiams.

$$F(\theta) = P(0 < \Theta \leq \theta), \quad 0 < \theta \leq 2\pi; \quad (1)$$

$$F(\theta + 2\pi) - F(\theta) = 1, \quad -\infty < \theta < \infty. \quad (2)$$

Lygybė (1) yra analogas ant apskritimo įprastos pasiskirstymo funkcijos ant tiesės. Lygybė (2) yra papildoma sąlyga, kuri atspindi kryptinių dydžių periodiškumą. Šią pasiskirstymo funkcijos savybę galima interpretuoti kaip teiginį, kad tikimybė patekti į tašką ant apskritimo gulinčio ant arkos, kurios ilgis yra 2π , yra 1.

Kryptinė pasiskirstymo funkcija turi savybes:

$$F(0) = 0, \quad F(2\pi) = 1.$$

Dar viena svarbi funkcija – yra kvantilių funkcija.

Kvantilių funkcija. Ši funkcija yra apibrėžiama u reikšmėms iš intervalo $[0, 1]$ kaip minimalus kampas θ , kuriame pasiskirstymo funkcijos reikšmė yra bent u :

$$Q(u) = \inf\{\theta : F(\theta) \geq u\}, \quad 0 \leq u \leq 1. \quad (3)$$

Kryptinė tikimybių tankio funkcija. Kaip ir tiesiniu atveju, jei kryptinė pasiskirstymo funkcija F yra absoliučiai tolydi, tada ji turi tankio funkciją f tokią, kad

$$\int_{\phi}^{\psi} f(\theta) d\theta = F(\psi) - F(\phi), \quad -\infty < \phi \leq \psi < \infty. \quad (4)$$

Funkcija f yra tankio funkcija tada ir tik tada, kai

1. $f(\theta) \geq 0$ beveik visur intervale $(-\infty, \infty)$,
2. $f(\theta + 2\pi) = f(\theta)$ beveik visur intervale $(-\infty, \infty)$,
3. $\int_0^{2\pi} f(\theta) d\theta = 1$.

Pirma savybė taip pat yra būdinga ir tiesiniams dydžiams, o antra ir trečia atsiranda dėl kryptinių dydžių periodiškumo.

Toliau yra nagrinėjamas kompleksinis atsitiktinis dydis $Z = e^{i\Theta}$. Apibrėžiame tokių dydžių charakteristinę funkciją, iš kurios galima gauti tiesinių momentų analogą kryptiniams dydžiams – trigonometrinius momentus.

Charakteristinė funkcija. Atsitiktinio kampo Θ charakteristinė funkcija apibrėžiama kaip kompleksinių skaičių seka $\{\tau_{p,0} : p = 0, \pm 1, \pm 2, \dots\}$, čia

$$\tau_{p,0} = E(e^{ip\Theta}) = \int_0^{2\pi} e^{ip\theta} dF(\theta), \quad p = 0, \pm 1, \pm 2, \dots \quad (5)$$

$\tau_{p,0}$ yra funkcijos F Fourier koeficientai. Jie yra vadinami p trigonometrinio Θ momentu apie nulinę kryptį ir gali būti užrašyti per kosinusų ir sinusų momentus:

$$\begin{aligned} \tau_{p,0} &= \alpha_p + i\beta_p, \\ \alpha_p &= E(\cos(p\Theta)) = \int_0^{2\pi} \cos(p\theta) dF(\theta), \\ \beta_p &= E(\sin(p\Theta)) = \int_0^{2\pi} \sin(p\theta) dF(\theta). \end{aligned} \quad (6)$$

Vidutinis vektorius. Vidutinis vektorius (angl. *mean resultant vector*) yra vienas iš parametrų nusakančių pasiskirstymo dėsnio savybes. Iš kiekvieno $\tau_{p,0}$ (5) yra išvedamas vidutinis vektorius, kurio ilgis yra ρ_p , o kryptis μ_p :

$$\rho_p = |\tau_{p,0}| = (\alpha_p^2 + \beta_p^2)^{1/2} \in [0, 1], \quad \mu_p = \text{atan2}(\beta_p, \alpha_p). \quad (7)$$

čia α_p ir β_p yra kosinusų ir sinusų momentai, o atan2 yra dviejų argumentų arktangento funkcija:

$$\text{atan2}(b, a) = \begin{cases} \arctan(b/a), & a > 0, \\ \arctan(b/a) + \pi, & b \geq 0, a < 0, \\ \arctan(b/a) - \pi, & b < 0, a < 0, \\ \pi/2, & b > 0, a = 0, \\ -\pi/2, & b < 0, a = 0, \\ \text{neapibrėžta}, & b = 0, a = 0. \end{cases} \quad (8)$$

Specialiu atveju, kai $p = 1$, gausime ρ_1 (arba ρ) ir μ_1 (arba μ), kurie yra pagrindiniai koncentracijos ir krypties matai, atitinkamai vadinami populiacijos vidutinis ilgis (angl. *mean resultant length*) ir populiacijos vidutinė kryptis (angl. *mean direction*).

Kryptiniai sklaidos matai. Yra svarbu apibrėžti sklaidos arba koncentracijos parametrus, kad galima būtų geriau nusakyti populiacijos pasiskirstymo savybes. Vienas iš koncentracijos matų, kuris buvo minėtas aukščiau (7), yra populiacijos vidutinis ilgis ρ . Vidutinis ilgis kinta intervale $[0, 1]$ ir įgija reikšmes artimas 1, kai taškai yra labai mažai nutolę nuo vidurinės krypties. ρ yra lygus nuliui, kai taškai yra išsidėstę tolygiai ant apskritimo arba kai pasiskirstymas yra multimodalus.

Populiacijos kryptinę dispersiją (angl. *circular variance*) galima apibrėžti:

$$v = 1 - \rho \in [0, 1]. \quad (9)$$

Kryptinis standartinis nuokrypis yra apibrėžiamas taip:

$$\sigma = (-2 \log \rho)^{1/2} \in [0, \infty]. \quad (10)$$

Kadangi į abi formules (9) ir (10) įeina ρ , taškų koncentracija galima nusakyti įvertinus tik vidutinį ilgį ir kitų matų skaičiuoti nebūtina.

Tokie sklaidos ant apskritimo matai nėra vieninteliai, bet jie yra dažniausiai naudojami.

Daugiamačiams kryptiniams dydžiams šios tikimybių teorijos sąvokas yra apibrėžiamos panašiai.

1.3. Aprašomoji statistika

Šitame skyriuje yra aprašyti imties analogai aukščiau apibrėžtiems kryptiniams populiacijos matams. Taip pat yra įvedami analogai ant apskritimo naudingiems tiesinės statistikos matams. Šiame skyriuje teorija yra aprašyta remiantys Mardia (1972); Mardia & Jupp (1999); Jammalamadaka & Sengupta (2001); Pewsey *et al.* (2013) darbais.

Tegul $\theta_1, \theta_2, \dots, \theta_n$ yra atsitiktinė kampų imtis, susiję su vektoriais $x_1, x_2, \dots, x_n$ ($x_j = (\cos \theta_j, \sin \theta_j)^T$) ir kompleksiniais skaičiais $z_1, z_2, \dots, z_n$ ($z_j = e^{i\theta_j}$).

Imties trigonometriniai momentai. Populiacijos trigonometrinių momentų išraiškoje (6) pakeitus teorinius vidurkius empiriniai gausime imties p eilės trigonometrinius momentus apie nulinę kryptį $t_{p,0}$.

$$\begin{aligned} t_{p,0} &= a_p + ib_p, \\ a_p &= \frac{1}{n} \sum_{j=1}^n \cos(p\theta_j), \quad b_p = \frac{1}{n} \sum_{j=1}^n \sin(p\theta_j), \end{aligned} \quad (11)$$

čia a_p ir b_p yra atitinkamai imties kosinusų ir sinusų momentai.

Imties vidutinis ilgis ir vidutinė kryptis. Analogiškai kaip ir populiacijos atveju iš trigonometrinių momentų yra gaunami vidutinio vektoriaus ilgis ir kryptis. Lygybėse (7) pakeitus teorinius momentus empiriniais gauname:

$$\bar{R}_p = (a_p^2 + b_p^2)^{1/2}, \quad (12)$$

$$\bar{\theta}_p = \text{atan2}(b_p, a_p). \quad (13)$$

Kai $p = 1$, gausime imties vidutinį ilgį $\bar{R}$ ir vidutinę kryptį $\bar{\theta}$.

1.3.1. Išplėtimas daugiamačiams kryptiniams dydžiams

Vidutinis vektorius sferiniams dydžiams. Tegul turime taškus $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ išdėstytus ant p -matės hipersferos. Tada jų poziciją galima apibendrinti vidutiniu vektoriumi erdvėje $\mathbb{R}^p$:

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i. \quad (14)$$

Kryptiniu atveju šį vektorių yra naudinga pateikti polinėje išraiškoje:

$$\bar{\mathbf{x}} = \bar{R} \bar{\mathbf{x}}_0, \quad (15)$$

čia vidutinis gautas ilgis $\bar{R} = \|\bar{\mathbf{x}}\|$, o vidutinė kryptis yra $\bar{\mathbf{x}}_0 = \|\bar{\mathbf{x}}\|^{-1} \bar{\mathbf{x}}$.

Jei $p = 2$, tai gausime atvejį, kai duomenys yra išdėstyti ant apskritimo. Tokiu atveju vidutinės krypties vektorius $\bar{\mathbf{x}}_0$ atitiks kampą $\bar{\theta}$, o vidutinio vektoriaus skaičiavimą galima suprastinti iki (13) ir (12) išraiškų.

Jei $p = 3$, t. y. atvejis, kai duomenys yra išdėstyti ant sferos, polines vidutinio vektoriaus $\bar{\mathbf{x}}_0 = (\bar{x}_0, \bar{y}_0, \bar{z}_0)$ koordinatės $(\bar{\theta}, \bar{\phi})$ galima gauti taip:

$$\begin{aligned} \bar{\theta} &= \arccos \bar{z}_0, \quad 0 \leq \bar{\theta} \leq \pi; \\ \bar{\phi} &= \text{atan2}(\bar{y}_0, \bar{x}_0), \quad 0 \leq \bar{\phi} < 2\pi. \end{aligned} \quad (16)$$

Vidutinis vektorius cilindriniais dydžiams. Tegul turime stebinių imtį $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$, kur $\mathbf{x}_i = (\theta_i, y_{1i}, y_{2i}, \dots, y_{ki})^T = (\theta_i, Y_i)^T$, išdėstyta ant k -mačio hipercilindro. Tada vidutinį vektorių ant sferos galima apibrėžti taip:

$$\boldsymbol{\mu} = (\bar{\theta}, \bar{Y})^T, \quad (17)$$

čia $\bar{\theta}$ yra skritulinio kintamojo vidutinė kryptis (13), o $\bar{Y}$ yra įprastas klasikinis vektoriaus vidurkis.

1.4. Pagrindiniai kryptiniai pasiskirstymai

Šitame skyriuje yra aprašomi pagrindiniai pasiskirstymai, kurie yra naudojami kryptinėje statistikoje. Kaip ir praeiti skyriai 1.2-1.3, šis skyrius parašytas remiantis Mardia (1972); Mardia & Jupp (1999); Jammalamadaka & Sengupta (2001) darbais.

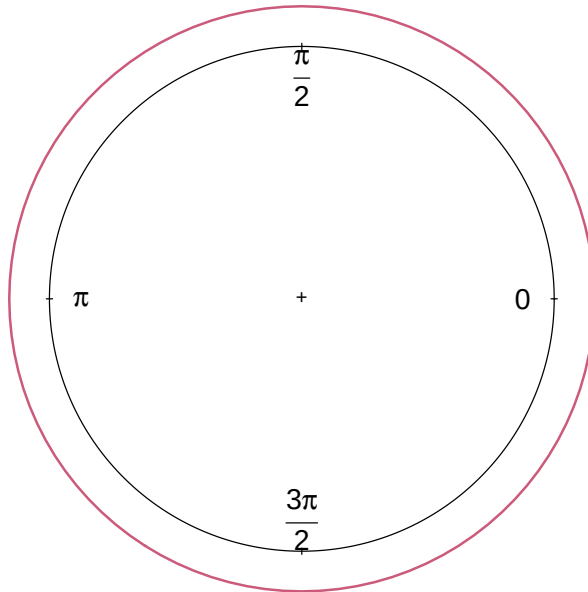
1.4.1. Vienmačiai pasiskirstymai

Kryptinis tolygusis pasiskirstymas. Jei atsitiktinis kampas yra iš kryptinio tolygaus pasiskirstymo, tai jo tankio funkcija yra:

$$f(\theta) = \frac{1}{2\pi}. \quad (18)$$

Tokio pasiskirstymo vidutinis ilgis $\rho = 0$, o vidutinė kryptis μ yra neapibrėžta.

Žemiau ant apskritimo yra pavaizduotas toks tankis.



5 pav. Kryptinio tolygiojo pasiskirstymo tankis

Kryptinis tolygusis pasiskirstymo dėsnis aprašo modelį ant apskritimo, kurio jokia kryptis nėra labiau tikėtina nei bet kuri kita. Jis apibūdina kryptinių duomenų izotropiją.

Susuktas normalusis pasiskirstymas. Susukant tiesinį normalųjį atsitiktinį dydį su vidurkiu $\xi \in (-\infty, \infty)$ ir dispersija $\sigma^2 > 0$, $X \sim \mathcal{N}(\xi, \sigma^2)$, aplink vienetinį apskritimą gauname kryptinį atsitiktinį dydį Θ , kuris yra iš susukto normalaus pasiskirstymo.

Tokio pasiskirstymo tankio funkcija yra:

$$f(\theta) = \frac{1}{(-4\pi \log \rho)^{1/2}} \sum_{k=-\infty}^{\infty} \exp \left\{ \frac{(\theta - \mu + 2\pi k)^2}{4 \log \rho} \right\}, \quad (19)$$

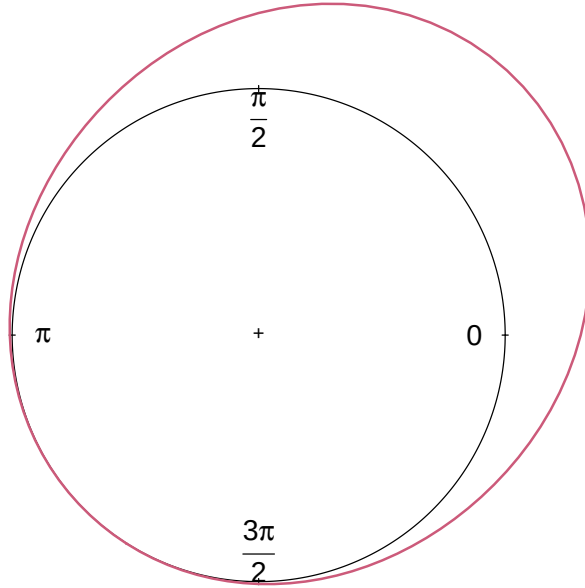
čia vidutinis ilgis ir vidutinė kryptis atitinkamai yra $\rho = e^{-\sigma^2/2}$ ir $\mu = \xi \bmod 2\pi$.

Šią tankio funkciją galima užrašyti ir kitu pavidalu:

$$f(\theta) = \frac{1}{2\pi} \left(1 + 2 \sum_{p=1}^{\infty} \rho^{p^2} \cos p(\theta - \mu) \right). \quad (20)$$

Praktikoje susuktas normalusis tankis pakankamai gerai aproksimuojamas per pirmus tris lygybės (20) narius, jei $\sigma^2 \geq 2\pi$, ir per lygybės (19) narį kai $k = 0$, jei $\sigma^2 \leq 2\pi$.

Susukto normalaus pasiskirstymo tankis yra pavaizduotas žemiau ant apskritimo.



6 pav. Susuktas normalusis pasiskirstymas, kai $\rho = 0.7$, o $\mu = \pi/4$

Šis pasiskirstymas yra unimodalus ir simetrinis apie vidutinę kryptį. Jis artėja prie kryptinio tolygaus, kai $\rho \rightarrow 0$.

Von Mises pasiskirstymas. Von Mises pasiskirstymas atlieka analogišką vaidmenį kaip ir normalusis (Gauso) pasiskirstymas tiesiniams dydžiams. Jis dažnai yra vadinamas normaliu pasiskirstymu ant apskritimo.

Šio pasiskirstymo tankio funkcija yra:

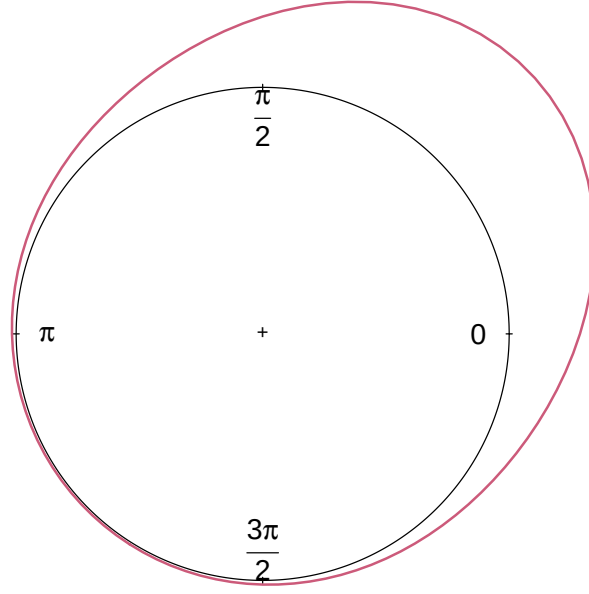
$$f(\theta) = \frac{1}{2\pi I_0(\kappa)} e^{\kappa \cos(\theta - \mu)}, \quad (21)$$

čia $I_p(\kappa)$ – p eilės modifikuota pirmojo tipo Bessel funkcija, μ – vidutinė kryptis ir κ – koncentracijos parametras. Vidutinis ilgis yra $\rho = A_1(\kappa) = \frac{I_1(\kappa)}{I_0(\kappa)}$.

Modifikuota pirmojo tipo Bessel funkcija turi pavidalą:

$$I_p(\kappa) = \frac{1}{2\pi} \int_0^{2\pi} \cos p\theta e^{\kappa \cos \theta} d\theta. \quad (22)$$

Von Mises pasiskirstymo tankio funkcija yra pavaizduota žemiau ant apskritimo.



7 pav. Von Mises pasiskirstymas, kai $\kappa = 2$, ($\rho \approx 0.7$), o $\mu = \pi/4$

Toks pasiskirstymas yra unimodalus ir simetrinis apie vidutinę kryptį. Kryptinis tolygusis pasiskirstymas gaunamas, kai koncentracijos parametras $\kappa = 0$. Be to, verta paminėti, kad susuktas normalusis pasiskirstymas pakankamai gerai aproksimuoja von Mises pasiskirstymą, kai vidutinis ilgis ρ nėra labai didelis.

Von Mises pasiskirstymo parametrus μ ir κ galima gauti, kaip įprastai, naudojant didžiausio tikėtimumo metodą. Gauti įvertiniai turi pavidalą:

$$\hat{\mu} = \bar{\theta}, \quad \hat{\kappa} = A^{-1}(\bar{R}), \quad (23)$$

čia $\bar{\theta}$ – empirinė vidutinė kryptis, $\bar{R}$ – empirinis vidutinis gautas ilgis, $A(x) = \frac{I_1(x)}{I_0(x)}$. Funkcijos $A(x)$ atvirkštinė neturi išreikštinio pavidalo, todėl jos reikšmės vertinamos naudojant iteracinius skaičiavimus.

Pirmą kartą von Mises pasiskirstymas buvo pasiūlytas von Mises (1918) tiriant išmatuotų atominių masių nukrypimus nuo bendrų reikšmių.

1.4.2. Daugiamatčiai pasiskirstymai

von Mises-Fisher pasiskirstymas Vienetinis atsitiktinis vektorius $\mathbf{x}$ (sferinis atsitiktinis dydis) yra pasiskirstęs pagal von Mises-Fisher (arba Langevin) pasiskirstymą $\mathcal{M}_p(\boldsymbol{\mu}, \kappa)$, jei jo tankio funkcija yra:

$$f(\mathbf{x}; \boldsymbol{\mu}, \kappa) = \left(\frac{\kappa}{2}\right)^{p/2-1} \frac{1}{\Gamma(p/2)I_{p/2-1}(\kappa)} \exp\{\kappa \boldsymbol{\mu}^T \mathbf{x}\}, \quad (24)$$

čia parametras $\boldsymbol{\mu}$ – vidutinė kryptis ($\|\boldsymbol{\mu}\| = 1$), κ – koncentracijos parametras, $I_v(\kappa)$ – v eilės modifikuota pirmojo tipo Bessel funkcija.

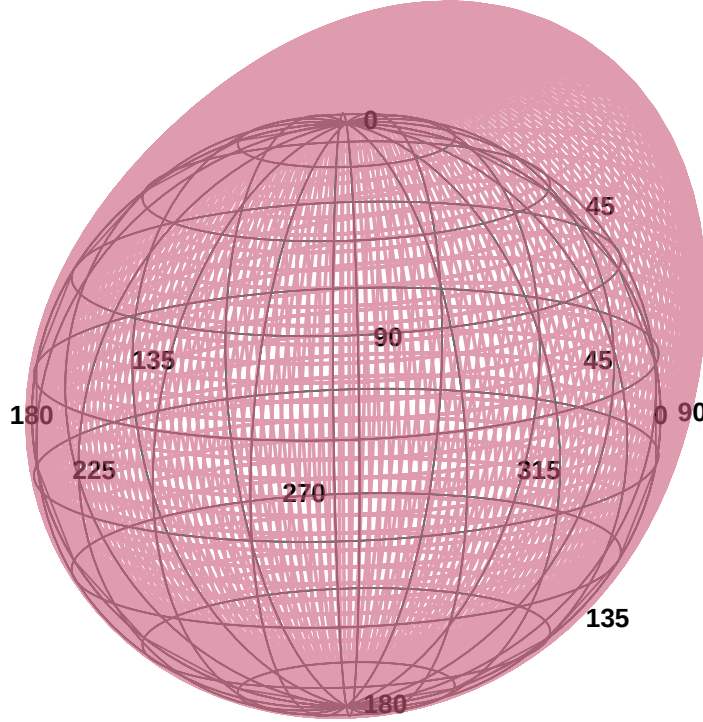
Tokio pasiskirstymo log-tankio funkcija yra tiesinė sferinio dydžio $\mathbf{x}$ atžvilgiu:

$$\log f(\mathbf{x}; \boldsymbol{\mu}, \kappa) = \kappa \boldsymbol{\mu}^T \mathbf{x} + \text{const}. \quad (25)$$

Atvejis, kai $p = 2$, gauname taškų išsidėsčiusių ant apskritimo von Mises pasiskirstymą (21). Jei $p = 3$, tai taškai yra išsidėstę ant sferos ir von Mises-Fisher pasiskirstymo tankio funkcija įgyja pavidalą:

$$f(\mathbf{x}; \boldsymbol{\mu}, \kappa) = \frac{\kappa}{\sinh \kappa} \exp \{ \kappa \boldsymbol{\mu}^T \mathbf{x} \}. \quad (26)$$

Tokiu atveju, kai $p = 3$, šį pasiskirstymą dažnai vadina tiesiog Fisher pasiskirstymu. Žemiau yra pavaizduotas tokio tankio pavyzdys:



8 pav. Von Mises-Fisher ant sferos ($p = 3$, vidutinė kryptis $\boldsymbol{\mu} = (\pi/4, \pi/3)$ ir koncentracijos parametras $\kappa = 4$)

Von Mises-Fisher pasiskirstymo parametrus $\boldsymbol{\mu}$ ir κ galima gauti, kaip įprastai, naudojant didžiausio tikėtimumo metodą. Gauti įvertiniai turi pavidalą:

$$\hat{\boldsymbol{\mu}} = \bar{\mathbf{x}}_0, \quad \hat{\kappa} = A_p^{-1}(\bar{R}), \quad (27)$$

čia $\bar{\mathbf{x}}_0$ – empirinis vidutinės krypties vektorius, $\bar{R}$ – empirinis vidutinis gautas ilgis, $A_p(x) = \frac{I_{p/2}(x)}{I_{p/2-1}(x)}$. Funkcijos $A_p(x)$ atvirkštinė neturi išreikštinio pavidalo, todėl jos reikšmės vertinamos naudojant iteracinius skaičiavimus.

Sferiniu dydžiu atveju, kai $p = 3$, funkcija $A_p(x)$ turi paprastesnį pavidalą:

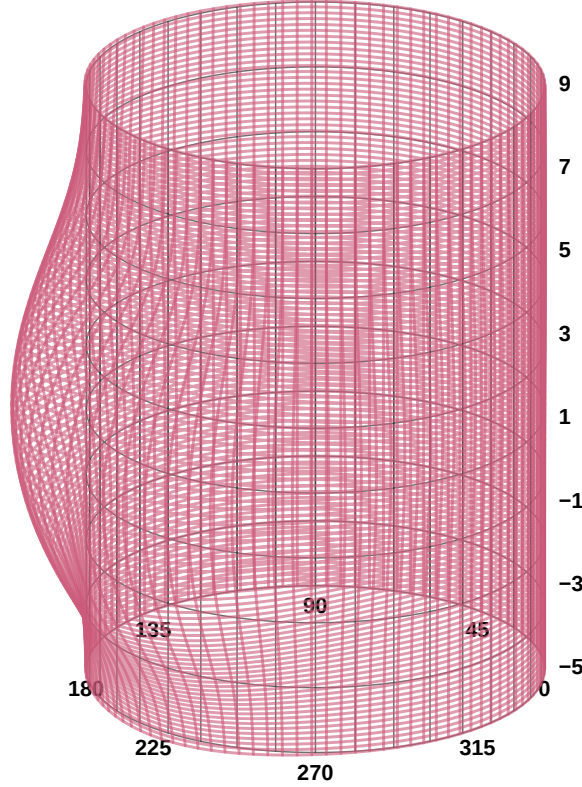
$$A_3(\kappa) = \coth x - \frac{1}{x}. \quad (28)$$

von Mises-Normalusis pasiskirstymas. Tegul turime cilindrinį atsitiktinį dydį (θ, y) , kur atsitiktinis dydis θ yra skritulinis, o y – vienmatis tiesinis. Mardia & Sutton (1978)

pasiūlė tokiam dydžiui aprašyti naudoti von Mises-Normalųjį pasiskirstymą. Šio pasiskirstymo $(\theta, y) \sim v\mathcal{MN}$ tankis atrodo taip:

$$f(y, \theta) = \frac{1}{(2\pi)^{\frac{3}{2}} I_0(\kappa) \sigma_c^2} \exp \left\{ \kappa \cos(\theta - \mu_0) - \frac{(y - \mu(\theta))^2}{2\sigma_c^2} \right\}. \quad (29)$$

Žemiau yra pavaizduotas tokio tankio pavyzdys:



9 pav. Von Mises-Normalusis tankis ant cilindro, vidutinė kryptis $\mu_0 = -3\pi/4$, koncentracijos parametras $\kappa = 3$, vidurkis $\mu = 4$, standartinis nuokrypis $\sigma = 3$, ryšio stiprumas $\rho = .6$ ir fazės kampas $\nu = \pi/3$

Nesunku pastebėti, kad marginaliniai atsitiktinių dydžių θ ir y pasiskirstymai yra von Mises ir sąlyginis normalusis:

$$\begin{aligned} \theta &\sim v\mathcal{M}(\mu_0, \kappa), \\ y|\theta &\sim \mathcal{N}(\mu(\theta), \sigma_c^2). \end{aligned} \quad (30)$$

Sąlyginio normaliojo pasiskirstymo parametrai yra lygus

$$\begin{aligned} \mu(\theta) &= \mu + \sigma\sqrt{\kappa}\rho \cos(\theta - \nu), \\ \sigma_c^2 &= \sigma^2 (1 - \rho^2). \end{aligned} \quad (31)$$

Tokiu būdu yra gaunamas pasiskirstymas su 6 parametrais $\mu_0, \kappa, \mu, \sigma, \rho$ ir ν .

Jei von Mises-Normaliojo pasiskirstymo parametras ρ , kuris nurodo ryšio stiprumą tarp cilindrinio dydžio komponentų, yra lygus 0, tai atsitiktiniai dydžiai θ ir y yra nepriklausomai pasiskirstę kaip von Mises $\theta \sim v\mathcal{M}(\mu_0, \kappa)$ ir normalusis $y \sim \mathcal{N}(\mu, \sigma^2)$.

Priešingu atveju atsitiktinio dydžio y marginalinis pasiskirstymas yra komplikotas pagal Mardia & Sutton (1978).

Von Mises-Normaliojo pasiskirstymo parametrus galima įvertinti didžiausio tikėtino metodo. Skritulinio kintamojo θ parametrų įverčiai sutampa vienmačio von Mises pasiskirstymo parametrų įverčiais (23):

$$\hat{\mu} = \bar{\theta}, \quad \hat{\kappa} = A^{-1}(\bar{R}). \quad (32)$$

Tiesinio kintamojo y parametrų vertinimas yra sudėtingesnis. Visų pirma galima pastebėti, kad sąlyginio vidurkio išraiškoje yra tiesinės-kryptinės regresijos modelis, kuris turi pavidalą

$$y = \mu + \gamma \cos(\theta - \nu) + \varepsilon. \quad (33)$$

Šią modelį galima užrašyti kitu pavidalu:

$$y = \mu + \beta_1 \cos \theta + \beta_2 \sin \theta + \varepsilon, \quad (34)$$

čia $\gamma = \sqrt{\beta_1^2 + \beta_2^2}$, o $\nu = \text{atan2}(\beta_2, \beta_1)$. Modelio parametrai įvertinami mažiausių kvadratų metodu, analogiškai kaip ir tiesinės regresijos atveju.

Parametro σ įvertis yra y kintamojo standartinis nuokrypis. Ryšio parametą ρ galima įvertinti iš

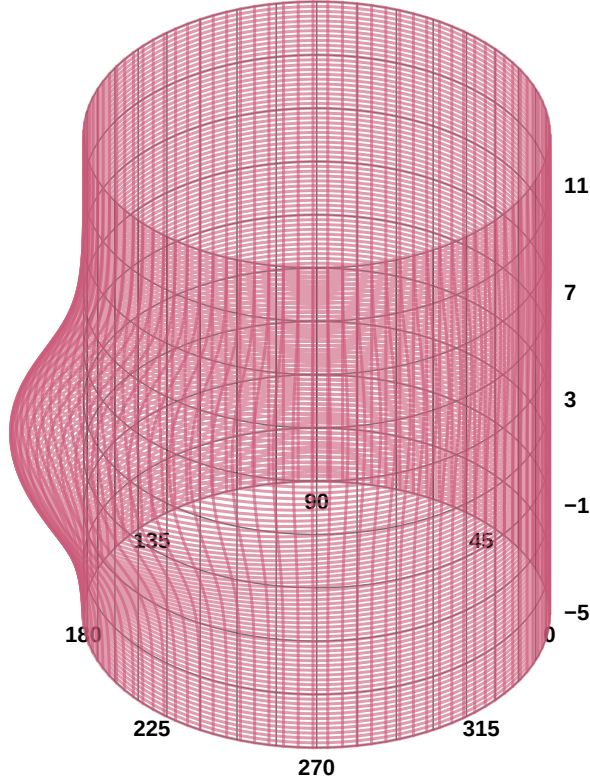
$$\gamma = \sigma \sqrt{\kappa} \rho \implies \rho = \frac{\gamma}{\sigma \sqrt{\kappa}}. \quad (35)$$

Dalinai susuktas normalusis pasiskirstymas. Kadangi aukščiau aprašytas cilindrinis von Mises-Normalusis pasiskirstymas yra limituotas tik tai vienam tiesiniam kintamajam ir yra komplikotas parametrų prasme, Roy *et al.* (2016) pasiūlė kitą pasiskirstymą cilindriniams arba hipercilindriniams dydžiams – pusiau susuktą normalųjį arba Gauso (*angl. semi-wrapped Gaussian*), žymimas *SWG*.

Tarkime turime atsitiktinį vektorių $(\mathbf{x} = (\theta, y) = (\theta, y_1, y_2, \dots, y_d))$ ant d -mačio hipercilindro. Pažymėkime $\mathbf{x} + 2w\pi$ vektorių $(\theta + 2w\pi, y_1, y_2, \dots, y_d)$. Tada SWG pasiskirstymo tankis užrašomas taip:

$$f(\mathbf{x}) = \sum_{w \in \mathbb{Z}} \mathcal{N}(\mathbf{x} + 2w\pi; \mu^{cl}, \Sigma^{cl}), \quad (36)$$

čia $\mathcal{N}(\mathbf{z})$ – daugiamatis normalusis tankis taške $\mathbf{z}$. Toks pasiskirstymas yra gaunamas susukant pirmąją vekatoriaus komponentę ant apskritimo, kai visos kitos lieka tiesinės. Žemiau yra pavaizduotas tokio tankio pavyzdys ant paprasto cilindro:



10 pav. Dalinai susuktas normalusis (SWG) tankis ant cilindro, vidutinis vektorius $\mu = (-3\pi/4, 4)$ ir kovariacijų matricos elementai $\sigma_1^2 = 0.5$, $\sigma_2^2 = 9$, $\sigma_{12} = 1.3$

SWG pasiskirstymo parametrų μ^{cl} ir Σ^{cl} įverčius galima gauti didžiausio tikėtinumų metodu. Kadangi analizinės išraiškos neegzistuoja parametrų vertinimas vyksta iteraciniu būdu. Yra naudojamas sekantis algoritmas:

1. pradiniam parametro μ^{cl} įverčiui priskiriamas cilindrinis vidutinis vektorius (17);
2. kiekvienam stebiniui $i = 1, 2, \dots, N$ randams nulinis dažnis w_0 , kuriam tiesinis atstumas nuo taško $\theta_i + 2w_0\pi$ iki vidutinio vektoriaus μ^{cl} pirmos komponentės yra mažiausias;
3. kiekvienam $i = 1, 2, \dots, N$ ir $w \in \mathbb{Z}$ apibrėžiamas dydis $\gamma_{wi} = 1$, jei $w = w_0$ ir $\gamma_{wi} = 0$ priešingu atveju;
4. pradinis parametro Σ^{cl} įvertis apskaičiuojamas pagal

$$\Sigma^{cl} = \frac{\sum_{i=1}^N \sum_{w \in \mathbb{Z}} \gamma_{wi} (\mathbf{x}_i + 2w\pi - \mu^{cl}) (\mathbf{x}_i + 2w\pi - \mu^{cl})^T}{N}; \quad (37)$$

5. atnaujiname reikšmes γ_{ij} , μ^{cl} ir Σ^{cl} :

(a) įvertiname naujas γ_{ij} reikšmes kiekvienam $i = 1, 2, \dots, N$ ir $w \in \mathbb{Z}$

$$\gamma_{wi} = \frac{\mathcal{N}(\mathbf{x} + 2w\pi; \mu^{cl}, \Sigma^{cl})}{\sum_{w \in \mathbb{Z}} \mathcal{N}(\mathbf{x} + 2w\pi; \mu^{cl}, \Sigma^{cl})}; \quad (38)$$

(b) perskaičiuojame Σ^{cl} pagal (37) ir μ^{cl} pagal

$$\mu^{cl} = \frac{\sum_{i=1}^N \sum_{w \in \mathbb{Z}} \gamma_{wi} (\mathbf{x}_i + 2w\pi)}{N}; \quad (39)$$

6. apskaičiuojame log-tikėtinumo reikšmę ir tikriname konvergavimą

$$\log L(\mu^{cl}, \Sigma^{cl} | \mathbf{x}) = \sum_{i=1}^N \log f(\mathbf{x}_i) = \sum_{i=1}^N \log \left(\sum_{w \in \mathbb{Z}} \mathcal{N}(\mathbf{x} + 2w\pi; \mu^{cl}, \Sigma^{cl}) \right); \quad (40)$$

7. paskutiniai du žingsniai kartojami tol kol bus pasiektas konvergavimas.

Atliekant visus skaičiavimus pasirenkamos 3 dažniu reikšmės $w = -1, 0, 1$, kadangi pagal Mardia & Jupp (1999) susuktas normalusis tankis gali būti aproksimuotas šiomis trimis komponentėmis.

1.5. Klasterizavimas kryptiniams duomenims

Šiame skyriuje yra nagrinėjami du kryptinių duomenų klasterizavimo būdai, kurie yra naudojami, kad suskirstyti duomenis į grupes pagal kintamųjų reikšmes. Jų tiesiniai analogai yra EM ir k-vidurkių klasterizavimo metodai. Tačiau klasikiniai algoritmai yra modifikuojami, kad atitiktų kryptinių duomenų savybes ir ypatumus.

1.5.1. EM klasterizavimas

EM algoritmas – tai lūkesčių-maksimizavimo (*angl. expectation-maximization*) algoritmas, kuris leidžia naudojant iteratyvų metodą surasti didžiausio tikėtinumo parametrų įverčius, kai modelis yra priklausomas nuo latentinių kintamųjų. Šis algoritmas vykdo du žingsnius: E (lūkesčių) ir M (maksimizavimo), vieną po kito, kol nebus surastas lokalus tikėtinumo funkcijos maksimumas.

Klasterizavimo uždavinio atveju, EM algoritmas yra taikomas, norint surasti labiausiai tikėtiną grupę, kai grupių pasiskirstymai yra nežinomi. Kitaip tariant, EM algoritmas bando įvertinti labiausiai tikėtiną mišinį skirtingų pasiskirstymų (*angl. mixture distribution*). Galutiniai klasteriai yra surandami pagal didžiausias priklausimo grupėms tikimybes.

Klasikinio EM klasterizavimo atveju yra daroma prielaida, kad duomenų skirstinys yra daugiamatis normalusis. Tokiu atveju mišinio pasiskirstymo log-tikėtinumo funkcija yra:

$$\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k) \right\}, \quad (41)$$

čia $\boldsymbol{\pi} = (\pi_1, \pi_2, \dots, \pi_K)$ – priklausimo klasėms tikimybės, $\boldsymbol{\mu} = (\mu_1, \mu_2, \dots, \mu_K)$ – klasių vidurkiai, $\boldsymbol{\Sigma} = (\Sigma_1, \Sigma_2, \dots, \Sigma_K)$ – klasių kovariacijų matricos.

Toliau galime apibrėžti EM klasterizavimo metodo algoritmą:

1. parenkamos pradinės grupių vidurkių $\boldsymbol{\mu}$ ir kovariacijų $\boldsymbol{\Sigma}$ reikšmės bei apriorinės priklausimo klasei tikimybės $\boldsymbol{\pi}$;
2. E žingsnis – perskaičiuojamos aposteriorinės kiekvieno taško priklausimo grupei tikimybės

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \mu_j, \Sigma_j)}; \quad (42)$$

3. M žingsnis – perskaičiuojami parametrai įverčiai: vidurkiai, kovariacijos, tikimybės:

$$\begin{aligned} \mu_k &= \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n, \\ \Sigma_k &= \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \mu_k) (\mathbf{x}_n - \mu_k)^T, \\ \pi_k &= \frac{N_k}{N}, \\ N_k &= \sum_{n=1}^N \gamma(z_{nk}); \end{aligned} \quad (43)$$

4. skaičiuojama log-tikėtinumo funkcija $\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ ir tikrinamas jos konvergavimas;
5. E ir M žingsniai kartojami kol nebus pasiektas konvergavimas.

Sferinis atvejis. Norint pritaikyti šį metodą sferiniams duomenims, reikia atsižvelgti į duomenų ypatumus: kintamieji yra periodiniai. Klasikinį EM klasterizavimo algoritmą reikia modifikuoti. (Banerjee *et al.*, 2005)

Visų pirma, daroma prielaida, kad duomenys yra pasiskirstę pagal von Mises-Fisher pasiskirstymą, t. y. kryptinį daugiamatį normalųjį, o ne tiesinį daugiamatį normalųjį. Vertinami parametrai: vidurkiai ir kovariacijų matricos, keičiami von Mises-Fisher pasiskirstymo parametrais: vidutinės krypties vektoriumi ir koncentracijos parametru.

Padarius tokius pakeitimus EM klasterizavimo algoritmas tampa sekančiu:

1. kiekvienas stebinyas priskiriamas prie atsitiktinės grupės;
2. naudojant didžiausio tikėtinumo metodą kiekvienai grupei įvertinami vidutinės krypties vektoriai μ_k ir koncentracijos parametrai κ_k (27).
3. E žingsnis – perskaičiuojamos aposteriorinės kiekvieno taško priklausimo grupei tikimybės γ_{ik} :

$$\gamma_{ik} = \frac{\pi_k \mathcal{M}(\mathbf{x}_i | \mu_k, \kappa_k)}{\sum_{j=1}^K \pi_j \mathcal{M}(\mathbf{x}_i | \mu_j, \kappa_j)}, \quad (44)$$

čia $\mathcal{M}(\mathbf{x}_i | \mu_k, \kappa_k)$ – von Mises-Fisher tankio funkcijos (24) reikšmė taške $\mathbf{x}_i$;

4. M žingsnis – perskaičiuojami parametų įverčiai:

$$\begin{aligned}\bar{\mathbf{x}}_k &= \frac{1}{N_k} \sum_{i=1}^n \gamma_{ik} \mathbf{x}_i, \quad \mu_k = \|\bar{\mathbf{x}}_k\|^{-1} \bar{\mathbf{x}}_k, \\ \bar{R}_k &= \|\bar{\mathbf{x}}_k\|, \quad \kappa_k = A_p^{-1}(\bar{R}_k) \\ \pi_k &= \frac{N_k}{N}, \quad N_k = \sum_{n=1}^N \gamma_{ik};\end{aligned}\tag{45}$$

5. skaičiuojama log-tikėtinumo funkcija $\ln f(\mathbf{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\kappa})$ ir tikrinamas jos konvergavimas

$$\ln f(\mathbf{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\kappa}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{M}_p(\mathbf{x}_n \mid \mu_k, \kappa_k) \right\};\tag{46}$$

6. E ir M žingsniai kartojami kol nebus pasiektas konvergavimas.

Tokį klasterizavimo algoritmą galima vadinti sferiniu EM klasterizavimu.

Cilindrinis atvejis. Norint taikyti EM klasterizavimo metodą cilindriniais duomenimis, klasikinį algoritmą reikia modifikuoti, kaip ir sferinių duomenų atveju. Šitoje vietoje pasinaudosime klasikinio metodo modifikavimo principu pritaikytų sferiniams duomenims ir analogiškai apibrėšime cilindrinį EM klasterizavimo analogą.

Visų pirma, kai stebiniai yra išsidėstę ant cilindro, EM klasterizavimui daroma prielaida, kad duomenys yra iš von Mises-Normaliojo skirstinio. Vertinami parametrai: vidurkiai ir kovariacijų matricos, keičiami į von Mises-Normaliojo pasiskirstymo 6 parametrus. Toliau galime apibrėžti šio metodo algoritmą:

1. kiekvienas stebinys priskiriamas prie atsitiktinės grupės;
2. naudojant didžiausio tikėtinumo metodą kiekvienai grupei įvertinami von Mises-Normaliojo pasiskirstymo parametrai: $\mu_{0k}, \kappa_k, \mu_k, \sigma_k, \rho_k$ ir ν_k (Skyrius 1.4.2);
3. E žingsnis – perskaičiuojamos aposteriorinės kiekvieno taško priklausimo grupei tikimybės γ_{ik} ;
4. M žingsnis – perskaičiuojami parametų įverčiai: $\mu_{0k}, \kappa_k, \mu_k, \sigma_k, \rho_k$ ir ν_k (aprašymas žemiau);
5. skaičiuojama log-tikėtinumo funkcija ir tikrinamas jos konvergavimas;
6. E ir M žingsniai kartojami kol nebus pasiektas konvergavimas.

Tokį EM klasterizavimo būdą vadinsime EMvMN (EM von Mises-Normaliuoju) klasterizavimo metodu.

M žingsnyje esančiam parametų perskaičiavimui standartinius didžiausio tikėtinumo įvertinius reikia pakeisti svoriniais įvertiniais, kaip daroma tiek klasikiniu atveju, tiek sferiniu.

Skritulinio kintamojo pasiskirstymo parametrai: vidutinė kryptis ir koncentracija, yra apskaičiuojami sekančiai:

$$\begin{aligned} C_k &= \frac{1}{N_k} \sum_{i=1}^N \gamma_{ik} \cos \theta_i, & S_k &= \frac{1}{N_k} \sum_{i=1}^N \gamma_{ik} \sin \theta_i, \\ \mu_{0k} = \bar{\theta}_k &= \text{atan2}(S_k, C_k), & \kappa_k &= A^{-1}(\bar{R}_k), & \bar{R}_k &= \sqrt{C_k^2 + S_k^2}, \\ N_k &= \sum_{i=1}^N \gamma_{ik}. \end{aligned} \quad (47)$$

Tiesinio kintamojo pasiskirstymo parametrus įvertinti, kiekvienai grupei k regresijos modelio (34) parametrų $\mu_k, \beta_{1k}, \beta_{2k}$ vertinimui yra naudojamas svorinis mažiausių kvadratų metodas. Fazės parametras ν_k yra gaunamas kaip ir anksčiau iš įvertintų regresijos parametrų $\nu = \text{atan2}(\beta_2, \beta_1)$. Standartinis nuokrypis σ_k yra vertinamas sekančiai:

$$\sigma_k^2 = \frac{1}{N_k - 1} \sum_{i=1}^N \gamma_{ik} (y_i - \bar{y})^2, \quad \bar{y} = \frac{1}{N_k} \sum_{i=1}^N \gamma_{ik} y_i. \quad (48)$$

Liko įvertinti ryšio parametą ρ_k , kuris yra gaunamas taip pat kaip ir anksčiau:

$$\rho = \sqrt{\beta_1 + \beta_2} / \sigma \sqrt{\kappa}. \quad (49)$$

Čia matrica $\gamma = (\gamma_{ik})$ yra sudaryta iš elementų

$$\gamma_{ik} = \frac{\pi_k f(\theta_i, y | \mu_{0k}, \kappa_k, \mu_k, \sigma_k, \rho_k, \nu_k)}{\sum_{j=1}^K \pi_j f(\theta_i, y | \mu_{0j}, \kappa_j, \mu_j, \sigma_j, \rho_j, \nu_j)}, \quad (50)$$

čia f – von Mises-Normaliojo dydžio tankis (29).

Kadangi prielaidos apie von Mises-Normalųjį pasiskirstymą naudojimas apriboja klasterizavimo algoritmą iki vienmačiu cilindrinų duomenų, tai Roy *et al.* (2016) pasiūlė EM cilindrinį klasterizavimą, vadinama SWGMM (*angl. Semi-Wrapped Gaussian Mixture Model*), tinkanti ir hipercilindriniais duomenimis.

Toks metodas vietoje klasikinio Gauso pasiskirstymo naudoja dalinai susuktą normalųjį (SWG). Vertinami parametrai išlieka vidurkiai ir kovariacijų matricos, tačiau kadangi pasiskirstymas skiriasi, parametrų vertinimas taip pat yra skirtingas. SWGMM modelio algoritmas yra apibrėžiamas sekančiai:

1. kiekvienas stebiny s priskiriamas prie atsitiktinės grupės;
2. naudojant iteracinį didžiausio tikėtimumo metodą kiekvienai grupei įvertinami vidurkiai ir kovariacijų matricos (Skyrius 1.4.2);
3. E žingsnis – perskaičiuojamos aposteriorinės kiekvieno taško priklausimo grupei tikimybės γ_{ik} ir kiekvieno taško paslėptos komponentės priklausimo grupei tikimybės γ_{ikw} ;
4. M žingsnis – perskaičiuojami vidurkių ir kovariacijų matricų įverčiai;
5. skaičiuojama log-tikėtimumo funkcija ir tikrinamas jos konvergavimas;

6. E ir M žingsniai kartojami kol nebus pasiektas konvergavimas.

E žingsnyje minėtos aposteriorinių tikimybių vertinimas yra vykdomas sekančiai:

- aposteriorinės kiekvieno taško priklausimo grupėms tikimybės yra įvertinamos pagal formulę:

$$\gamma_{ik} = \frac{\pi_k f(\mathbf{x}_i | \mu_k^{cl}, \Sigma_k^{cl})}{\sum_{j=1}^K \pi_j f(\mathbf{x}_i | \mu_j^{cl}, \Sigma_j^{cl})}, \quad (51)$$

čia $f(\mathbf{x}_i | \mu_k^{cl}, \Sigma_k^{cl})$ – SWG tankis (36).

- aposteriorinės kiekvieno taško paslėptų komponentų priklausimo grupėms tikimybės yra

$$\gamma_{ikw} = \frac{\pi_k \mathcal{N}(\mathbf{x} + 2w\pi; \mu_k^{cl}, \Sigma_k^{cl})}{\sum_{j=1}^K \pi_j f(\mathbf{x}_i | \mu_j^{cl}, \Sigma_j^{cl})}, \quad (52)$$

čia $\mathcal{N}$ – daugiamačio normaliojo dydžio tankis.

M žingsnyje esantys parametrai yra vertinami pagal sekančias formules:

$$\mu_k^{cl} = \frac{\sum_{i=1}^N \sum_{w \in \mathbb{Z}} \gamma_{ikw} (\mathbf{x}_i + 2w\pi)}{\sum_{i=1}^N \sum_{w \in \mathbb{Z}} \gamma_{ikw}}; \quad (53)$$

$$\Sigma_k^{cl} = \frac{\sum_{i=1}^N \sum_{w \in \mathbb{Z}} \gamma_{ikw} (\mathbf{x}_i + 2w\pi - \mu_k^{cl}) (\mathbf{x}_i + 2w\pi - \mu_k^{cl})^T}{\sum_{i=1}^N \sum_{w \in \mathbb{Z}} \gamma_{ikw}}. \quad (54)$$

1.5.2. k-vidurkių klasterizavimo metodas

k-vidurkių (*angl. k-means*) klasterizavimas – tai toks duomenų skaidymo į grupes būdas, kurio metu stebiniai yra priskiriami prie arčiausiai esančios grupės. Kitaip tariant tokio klasterizavimo metu yra sprendžiamas dispersijos minimizavimo grupėse uždavinys iteraciniu būdu.

Tegul turime N stebinių $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ iš K grupių. Apibrėžkime indikatorių $r_{nk} \in \{0, 1\}$, kuris yra lygus 1, jei stebinys n priklauso grupei k , ir lygus 0, priešingu atveju, čia $n = 1, \dots, N$ ir $k = 1, \dots, K$. Tada k-vidurkių klasterizavimo algoritmas bando minimizuoti funkciją J :

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \mu_k\|^2 \quad (55)$$

Klasikiniu, tiesinių duomenų atveju k-vidurkių klasterizavimo algoritmo žingsniai tokie:

1. pasirenkamas grupių skaičius K , taškai atsitiktinai priskiriami grupėms;

2. suskaičiuojami grupių vidurkiai

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^N r_{nk} \mathbf{x}_n}{\sum_{n=1}^N r_{nk}}; \quad (56)$$

3. įvertinami kiekvieno taško Euklidiniai atstumai iki grupių centrų (vidurkių) $\|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2$;

4. taškai priskiriami prie arčiausios grupės

$$r_{nk} = \begin{cases} 1, & \text{jei } k = \arg \min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2, \\ 0, & \text{priešingu atveju;} \end{cases} \quad (57)$$

5. iteracijos kartojamos iki konvergavimo, t. y. kol grupės nebesikeičia.

Palyginus k-vidurkių klasterizavimo metodą su EM klasterizavimu, šis metodas nereikalauja jokių prielaidų apie pasiskirstymo dėsnį. Taip pat galima pastebėti, kad k-vidurkių algoritmą galima suskaidyti į dvi fazes: funkcijos J minimizavimas pagal r_{nk} , kai grupių vidurkiai $\boldsymbol{\mu}_k$ yra fiksuoti ir funkcijos J minimizavimas pagal $\boldsymbol{\mu}_k$, kai indikatoriaus reikšmės r_{nk} yra fiksuotos. Šios dvi fazės atitinka EM algoritmo E (1 fazė) ir M (2 fazė) žingsnius.

Tokiu būdu k-vidurkių klasterizavimą galima laikyti EM klasterizavimo atskiru atveju, kai yra daromas griežtas klasterių priskyrimas ir duomenų kovariacijų matrica yra lygi $\varepsilon \mathbf{I}$, čia ε – grupių dispersija.

Sferinis atvejis. Kaip ir EM metodo atveju, k-vidurkių metodui taikyti sferiniams duomenims reikia modifikavimų. Kad algoritmas atsižvelgtų į periodinę duomenų struktūrą, keičiamas klasterių centrų skaičiavimas bei taškų atstumų matas. (Kesemen *et al.*, 2016) Tada sferinis k-vidurkių algoritmas yra:

1. pasirenkamas grupių skaičius k , taškai atsitiktinai priskiriami grupėms;
2. vertinami grupių vidutinės krypties vektoriai $\bar{\mathbf{x}}_0$;
3. įvertinami kiekvieno taško lanko atstumai iki grupių centrų: $\arccos(\bar{\mathbf{x}}_0^T \mathbf{x})$;
4. taškai priskiriami prie arčiausios grupės;
5. iteracijos kartojamos iki konvergavimo, t. y. kol grupės nebesikeičia.

Cilindrinis atvejis. Norint k-vidurkių klasterizavimo metodą pritaikyti kito tipo daugiačiams kryptiniams duomenims – cilindriniais dydžiams, klasikinį algoritmą taip pat reikia modifikuoti, tačiau pakeitimai nėra identiški sferiniam atvejui.

Kad gautume k-vidurkių algoritmą, kuris atsižvelgtų į cilindrinį duomenų struktūrą, reikia pakeisti klasterių centrų skaičiavimą bei taškų atstumų matą, kaip ir sferiniu atveju. (Fujita, 2017) Centrų skaičiavimui yra naudojamas vidutinis vektorius ant cilindro, o vietoje Euklidinio atstumo tarp taškų yra naudojamas cilindrinis dydžių panašumo matas, kuris yra apibrėžiamas sekančiai:

$$S(\mathbf{x}, \mathbf{m}) = \kappa \cos(\theta_x - \theta_m) - \frac{(y_x - y_m)^2}{2\sigma^2}, \quad (58)$$

čia $\mathbf{x} = (\theta_x, y_x)$, $\mathbf{m} = (\theta_m, y_m)$ – du cilindriniai taškai, κ – skritulinio kintamojo koncentracija, σ^2 – tiesinio kintamojo dispersija. Tada cilindrinis k-vidurkių arba cyk-vidurkių algoritmas yra:

1. pasirenkamas grupių skaičius k, taškai atsitiktinai priskiriami grupėms;
2. vertinami grupių vidutiniai vektoriai $\boldsymbol{\mu}$ pagal (17), koncentracijos parametrai κ ir dispersijos σ^2 ;
3. įvertinami kiekvieno taško ir grupių centrų cilindriniai panašumai $S(\mathbf{x}, \mathbf{m})$;
4. taškai priskiriami prie arčiausios grupės;
5. iteracijos kartojamos iki konvergavimo, t. y. kol grupės nebesikeičia.

Kadangi cyk-vidurkių algoritmas turi daug parametrų, kuriuos tenka vertinti kiekvienoje iteracijoje, Fujita (2017) pasiūlė šį algoritmą suprastinti fiksuojant kelis parametrus – koncentracijas ir dispersijas. Tokį algoritmą galima pavadinti fiksuotų cyk-vidurkių klasterizavimu.

Koncentracijos parametro ir dispersijos fiksuotos reikšmės natūraliai seka iš cilindrinio atstumo mato, kuris yra

$$D(\mathbf{x}, \mathbf{m}) = 1 - \cos(\theta_x - \theta_m) + \|y_x - y_m\|, \quad (59)$$

čia kiekvieno tiesinio kintamojo y_k dispersijos turi būti lygios 1. Iš čia seka, kad koncentracijos parametras galima pasirinkti $\kappa = 1$, o dispersiją σ^2 lygią tiesinio kintamojo visos imties dispersijai. Tokiu būdu cyk-means algoritmas tampa:

1. pasirenkamas grupių skaičius k, taškai atsitiktinai priskiriami grupėms, tiesiniai kintamieji yra standartizuojami, kad jų dispersija būtų lygi 1;
2. vertinami grupių vidutiniai vektoriai $\boldsymbol{\mu}$ pagal (17);
3. įvertinami kiekvieno taško cilindrinis atstumas iki grupių centrų $D(\mathbf{x}, \mathbf{m})$;
4. taškai priskiriami prie arčiausios grupės;
5. iteracijos kartojamos iki konvergavimo, t. y. kol grupės nebesikeičia.

1.6. Kryptinių duomenų klasifikavimas

Šiame skyriuje yra nagrinėjami kryptinių duomenų klasifikavimo būdai. Priešingai, nei aukščiau aprašytuose klasterizavimo metoduose, klasifikavimo atveju tikros grupės yra iš anksto žinomos ir tikslas yra „išmokti“ jas prognozuoti. Toliau yra aprašyti keli kryptinių duomenų klasifikavimo algoritmai, kurių tiesiniai analogai yra diskriminantinė analizė ir logistinė regresija. Kaip ir klasterizavimo atveju, klasikiniai klasifikavimo algoritmai taip pat yra modifikuojami, kad atitiktų kryptinių duomenų savybes ir ypatumus.

1.6.1. Diskriminantinė analizė

Diskriminantinė analizė – tai toks klasifikavimo metodas, kuris sukonstruoja diskriminavimo taisyklę arba ribą, kuri suskirsto stebinius į grupes.

Klasikinė diskriminantinė analizė yra paremta Bajeso statistika, t. y. vertinamos aposteriorinės tikimybės priklausyti kiekvienai klasei ir kiekvienam stebiniui yra priskiriama labiausiai tikėtina grupė.

Tarkime turime atsako kintamąjį y , kuris įgyja reikšmes $y \in \{1, 2, \dots, K\}$ (čia K yra klasių skaičius) ir prediktorius X . Tarkime duomenų pasiskirstymas yra $f_k(X)$, $k = 1, 2, \dots, K$, o apriorinės priklausymo klasei tikimybės π_k , $k = 1, 2, \dots, K$. Tada aposteriorinė tikimybė yra

$$P(y = k | X = x) = \frac{\pi_k f_k(x)}{\sum_{i=1}^K \pi_i f_i(x)}. \quad (60)$$

Atlikus kelis pertvarkymus gauname diskriminantinę funkciją, kurią reikia maksimizuoti pagal visas grupes, norint gauti grupių prognozes.

$$\delta_k(X) = \ln \pi_k + \ln f_k(X) \longrightarrow \max_k \quad (61)$$

Sferinis atvejis. Sferinių/hipersferinių duomenų atveju, daroma prielaida, kad duomenys turi von Mises-Fisher pasiskirstymą (24), t. y. sferinį normalųjį pasiskirstymą, vietoje Gauso pasiskirstymo, kaip yra klasikinėje diskriminantinėje analizėje (Figueiredo, 2009). Įstačius į formulę (61) von Mises-Fisher tankį gauname

$$\delta_k(x) = \ln \pi_k + \ln C_p(\kappa_k) + \ln \kappa \mu_k^T x \longrightarrow \max_k, \quad (62)$$

čia $C_p(\kappa_k)$ – normavimo konstanta, p – erdvės dydis, κ_k – k klasės koncentracijos parametras, μ_k – k klasės vidutinė kryptis, x – sferinių dydžių koordinatės.

Kai $p = 3$, t. y. duomenys yra išdėstyti ant įprastos sferos, diskriminantinę funkciją galima užrašyti taip:

$$\delta_k(x) = \ln \pi_k + \ln \kappa_k - \ln \sinh \kappa_k + \ln \kappa \mu_k^T x \longrightarrow \max_k, \quad (63)$$

Kadangi, įprastai, parametrų reikšmės yra nežinomos, jos yra vertinamos iš duomenų naudojant didžiausio tikėtimumo įvertčius kiekvienai grupei. Von Mises-Fisher pasiskirstymo parametrai gali būti įvertinti pasinaudojus formule (27). Apriorinės tikimybės π_k yra parenkamos lygios grupių santykiniais dažniais.

Toks sferinis klasifikavimo algoritmas yra analogas tiesinės diskriminantinės analizės. Nors tiesiniu klasikiniu atveju yra daroma papildoma prielaida, kad grupių dispersijos yra lygios, sferiniu atveju koncentracijos parametrai gali skirtis tarp grupių, kadangi von Mises-Fisher pasiskirstymas ir taip uždeda papildomų apribojimų duomenims.

Taip pat, verta paminėti, kad toks klasifikavimo algoritmas nesugebės gerai suklasifikuoti duomenų, jei duomenų skirstinys nėra panašus į von Mises-Fisher skirstinį. Tokiu atveju galima diskriminantinės analizės algoritme von Mises-Fisher pasiskirstymą pakeisti kitu sferiniu pasiskirstymu arba naudoti kitą sferinį klasifikavimo būdą. (Tsagris & Alenazi, 2019)

Cilindrinis atvejis. Kad gautume diskriminantinės analizės analogą cilindriniais duomenims, pasinaudojame sferinių pakeitimų pavyzdžiu ir pakeičiama prielaidą apie pasiskirstymą. Cilindriniais dydžiams vietoje Gauso pasiskirstymo naudosime von Mises-Normalųjį pasiskirstymą. Įstačius į diskriminantinę funkciją (61) atitinkama von Mises-Normalųjį tankį (29) gauname

$$\delta_k(\theta, y) = \ln \pi_k - \ln I_0(\kappa) - \ln \sigma_c^2 + \kappa \cos(\theta - \mu_0) - \frac{(y - \mu(\theta))^2}{2\sigma_c^2} \longrightarrow \max_k. \quad (64)$$

Kadangi įprastai parametrų reikšmės yra nežinomas, jos yra vertinamos iš duomenų naudojant didžiausio tikėtino įverčius kiekvienai grupei. Von Mises-Normaliojo pasiskirstymo parametrai gali būti įvertinti didžiausio tikėtino metodo, kaip aprašyta skyriuje 1.4.2. Apriorinės tikimybės π_k yra parenkamos lygios grupių santykiniais dažniams, kaip ir klasikiniu bei sferiniu atveju.

Tokį cilindrinį klasifikavimo algoritmą galima pavadinti *von Mises-Normaliąja diskriminantine analize*. Ji yra analogas kvadratinės diskriminantinės analizės, nes tiesinės klasikinės diskriminantinės analizės atveju yra daroma papildoma prielaida, kad grupių dispersijos yra lygios. Dėl von Mises-Normaliojo skirstinio pavidalo panašų apribojimą sunku įvesti, todėl yra nagrinėjamas tik kvadratinės diskriminantinės analizės analogas.

Taip pat verta paminėti, kad toks klasifikavimo būdas tinka tik, jei yra vienas tiesinis kintamasis. Be to, toks algoritmas nesugebės gerai suklasifikuoti duomenų, jei duomenų skirstinys nėra panašus į von Mises-Normalųjį.

Kadangi von Mises-Normalioji diskriminantinė analizė yra apribota ant vienmačio cilindro, verta nagrinėti kitą cilindrinį pasiskirstymą.

Tegul turime hipercilindrinis duomenis. Tada diskriminantinėje analizėje pakeičiame prielaidą apie pasiskirstymą iš Gauso į dalinai susuktą normalųjį (SWG). Įstačius į diskriminantinę funkciją (61) SWG tankį (36) gauname

$$\delta_k(\theta, y) = \ln \pi_k + \ln \left(\sum_{w \in \mathbb{Z}} \mathcal{N}(\mathbf{x} + 2w\pi; \mu^{cl}, \Sigma^{cl}) \right) \longrightarrow \max_k. \quad (65)$$

Nežinomus pasiskirstymo parametrus, kaip ir anksčiau, galima įvertinti didžiausio tikėtino metodo. SWG parametrų μ^{cl} ir Σ^{cl} vertinimas yra aprašytas skyriuje 1.4.2. Apriorinės tikimybės π_k yra parenkamos lygios grupių santykiniais dažniams, kaip ir visais kitais atvejais.

Šį hipercilindrinis duomenų klasifikavimo būdą galima vadinti *SWG diskriminantine analize*. Jeigu įvesime apribojimą, kad kovariacijų matricos yra lygios visuose grupėse, gausime tiesinės diskriminantinės analizės analogą, priešingu atveju tai yra kvadratinės diskriminantinės analizės analogas.

Verta paminėti, kad šiame algoritme grupių pasiskirstymų parametrų vertinimas yra iteracinis procesas, todėl jei grupių arba stebinių yra daug šis klasifikavimo metodas bus lėtas.

1.6.2. Logistinė regresija

Logistinė regresija – tai toks apibendrintas tiesinis modelis, kuris naudojamas tikimybių priklausyti vienai iš dviejų klasių prognozavimui.

Tegul turime prediktorius X ir atsako kintamąjį $Y \in \{0, 1\}$. Pažymėkime tikimybę atsakui priklausyti pirmai klasei $p = P(Y = 1|X)$. Tada apibendrintas tiesinis modelis turės sekančią išraišką

$$g(p) = \beta_0 + X\beta, \quad (66)$$

čia $g(p)$ – sąryšio funkcija. Logistinės regresijos atvejų kaip sąryšio funkcija yra naudojama logit funkcija.

$$\text{logit}(p) = \ln \frac{p}{1-p} = \beta_0 + X\beta. \quad (67)$$

Atsako kintamojo prognozės gaunamos lyginant įvertintas tikimybes p su tam tikra ribinė klasių atskyrimo tikimybe, kuri dažniausiai yra parenkama lygi 0,5.

Jei atsako kintamasis turi daugiau nei dvi skirtingas klases, tarkime k , tai sudaromas multinominės logistinės regresijos modelis, kuris susidaro iš $k - 1$ modelių, kurie vertina kiekvienos klasės tikimybes atskirai. Likusios klasės k tikimybė yra išreiškiama per įvertintas tikimybes.

Sferinis atvejis. Norint tokio tipo modelį pritaikyti sferiniams duomenims, reikia jį modifikuoti, kad jis atitiktų sferinių kintamųjų ypatumus.

Iš pradžių, prieš pereinant prie daugiamachių kryptinių duomenų, apibrėžiame logistinės regresijos modelį vienmačiams kryptiniams duomenims, t. y. ant apskritimo.

Tarkime vietoje prediktorių X turime skritulinį kintamąjį θ . Tada pagal Al-Daffaie & Khan (2017) skritulinės logistinės regresijos modelis yra apibrėžiamas taip:

$$\text{logit}(p) = \ln \frac{p}{1-p} = \beta_0 + \beta_1 \cos \theta + \beta_2 \sin \theta. \quad (68)$$

Čia verta paminėti, kad kampas θ Dekarto koordinačių sistemoje atitinka vektorių $\mathbf{x} = (\cos \theta, \sin \theta)$. Dėl to skritulinės logistinės regresijos modelį galima užrašyti ir kitokiu pavidalu:

$$\text{logit}(p) = \ln \frac{p}{1-p} = \beta_0 + \mathbf{x}\beta. \quad (69)$$

Toliau pagal skritulinės logistinės regresijos pavyzdį yra apibrėžiamas sferinės logistinės regresijos modelis.

Tarkime turime sferinį kintamąjį (θ, ϕ) kaip prediktorių. Toks kintamasis atitinka vektorių $\mathbf{x} = (\sin \theta \cos \phi, \sin \theta \sin \phi, \cos \theta)$. Kaip ir vienmačio kryptinio kintamojo atvejų, taip ir daugiamachio, logistinės regresijos modelis turės pavidalą (69), pasikeis tik vektoriaus $\mathbf{x}$ dydis bei koordinatės. Taigi sferinės logistinės regresijos modelis yra apibrėžiamas sekančiai

$$\text{logit}(p) = \ln \frac{p}{1-p} = \beta_0 + \beta_1 \sin \theta \cos \phi + \beta_2 \sin \theta \sin \phi + \beta_3 \cos \theta. \quad (70)$$

Sferinės logistinės regresijos modelį galima apibendrinti, pridėdant tiesinių kintamųjų, kaip Fernandes & Cardoso (2016) padarė skritulinių kintamųjų atveju.

Tegul dabar kaip prediktorius turime sferinį kintamąjį (θ, ϕ) ir tiesinius kintamuosius X . Tada hibridinį sferinės logistinės regresijos modelį galima apibrėžti taip

$$\text{logit}(p) = \ln \frac{p}{1-p} = \beta_0 + \beta_1 \sin \theta \cos \phi + \beta_2 \sin \theta \sin \phi + \beta_3 \cos \theta + X\beta. \quad (71)$$

Cilindrinis atvejis. Analogiškai, kaip ir sferinių duomenų atveju, logistinę regresiją galima apibrėžti ir ant cilindro.

Tegul turime du kintamuosius: skritulinį θ ir tiesinį y , kurie kartu paėmus sudaro cilindrinį kintamąjį. Dekarto koordinačių sistemoje toks cilindrinis kintamasis atitinka vektorių $\mathbf{x} = (\cos \theta, \sin \theta, y)$. Tada cilindrinės logistinės regresijos modelį galima apibrėžti taip:

$$\text{logit}(p) = \ln \frac{p}{1-p} = \beta_0 + \beta_1 \cos \theta + \beta_2 \sin \theta + \beta_3 y. \quad (72)$$

Kaip ir sferinės logistinės regresijos atveju, cilindrinę logistinę regresiją galima apibendrinti pridėjus daugiau kintamųjų. Jei vietoje vieno tiesinio kintamojo y pasiimsime tiesinių kintamųjų matricą X , tai hipercilindrinės logistinės regresijos modelį galima apibrėžti sekančiai:

$$\text{logit}(p) = \ln \frac{p}{1-p} = \beta_0 + \beta_1 \cos \theta + \beta_2 \sin \theta + X\beta. \quad (73)$$

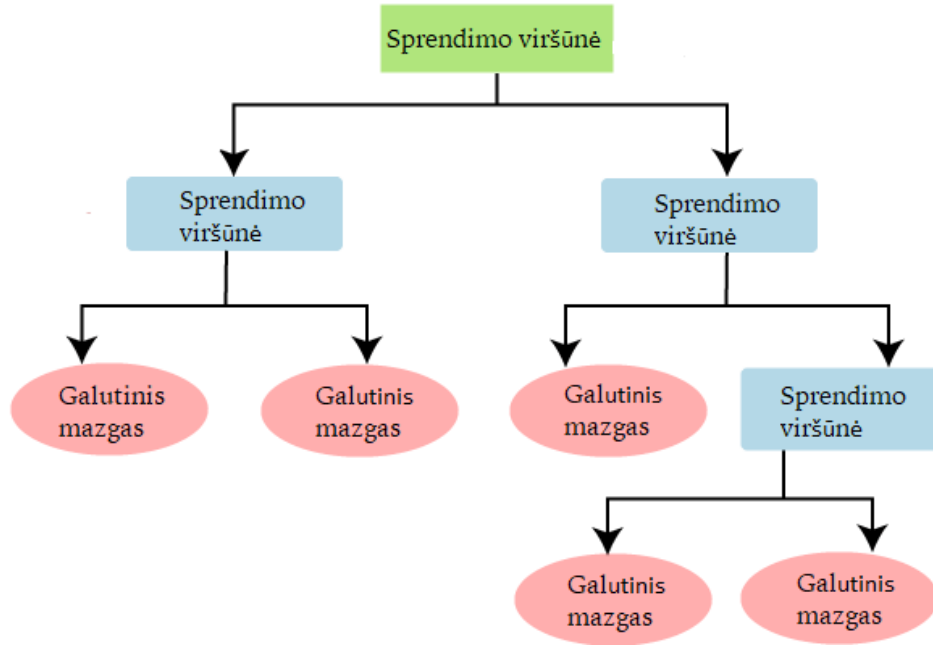
Kadangi visais kryptinės logistinės regresijos atvejais modeliai išlieka log-tiesiniai parametru atžvilgiu, modelio parametrus galime įvertinti taip pat kaip ir klasikinės logistinės regresijos atveju.

1.7. Kryptinių duomenų sprendimų medžiai

Šiame skyriuje yra aprašomas sprendimų medžių veikimo principas ir jų pritaikymas kryptiniams duomenims, o būtent kryptinių medžių von Mises modelis, pasiūlytas (Lang *et al.*, 2020a).

Sprendimų medžio modelio veikimo principas remiasi sekančių algoritmu:

1. pasirenkamos visos duomenų viršūnės (pradedant nuo visų duomenų), kurios nėra galutiniai mazgai (viršūnės), ir kiekviena tokia viršūnė, dalinama į dvi dalis (naujas viršūnės) pagal tam tikrą taisyklę;
2. viršūnės dalinamos iki tol, kol tenkinamas tam tikras stabdymo kriterijus;
3. kiekvienoje galutinėje viršūnėje atsako kintamojo prognozei priskiriama tam tikra reikšmė.



11 pav. Sprendimų medžio modelio schema

Dabar reikia apibrėžti minėtas taisykles, kriterijus ir prognozių reikšmes, kad juos atitiktų kryptinių duomenų savybes. Gautas modelis yra – *kryptinių medžių von Mises modelis*. Šis modelis kampinį kintamąjį suskaido į grupes pagal papildomus kintamuosius ir kiekvienai grupei parenka von Mises pasiskirstymą.

Kryptinių medžių von Mises modelio algoritmas:

- Kiekvienoje viršūnėje (pradedant nuo visų duomenų) įvertinami von Mises pasiskirstymo parametrai μ ir κ ir apskaičiuojami kiekvieno stebinio indėliai (*angl. score function*):

$$\begin{aligned}
 s(\mu, \kappa, \theta) &= \left(\frac{\partial \ell}{\partial \mu}(\mu, \kappa; \theta), \quad \frac{\partial \ell}{\partial \kappa}(\mu, \kappa; \theta) \right) \\
 &= \left(\kappa \sin(\theta - \mu), \quad -\frac{I_1(\kappa)}{2\pi I_0(\kappa)} + \cos(\theta - \mu) \right),
 \end{aligned} \tag{74}$$

čia ℓ von Mises logtikėtimumo funkcija.

- Tikrinama nepriklausomumo hipotezė tarp papildomų kintamųjų z_j ir indėlių, dalijimo kintamasis – kintamasis su mažiausia p-reikšme. Dalijimas stabdomas, kai hipotezė yra priimama.
- Dalijimosi taškas r yra parenkamas taip, kad neatitikimas tarp dviejų grupių $\mathcal{B}_{1r}$ ir $\mathcal{B}_{2r}$ būtų didžiausias, t. y. mažiausias indėlių sumas:

$$t^{qr} = \sum_{i \in \mathcal{B}_{qr}} s(\hat{\mu}, \hat{\kappa}, \theta_i) \tag{75}$$

- Kiekvienoje nedalomoje viršūnėje parenkamas von Mises pasiskirstymas.

Kryptiniai sprendimų medžiai gali būti taikomi, kad įvertinti kryptinio kintamojo pasiskirstymą, kai jis priklauso nuo kitų papildomų kintamųjų. Taip pat sprendimų medžiai gali padėti atrinkti prediktorius, kurie yra labiausiai susiję su kryptiniu atsako kintamuoju, arba pašalinti nepriklausomus. Be to, kryptiniai sprendimų medžiai gali būti naudojami naujų stebinių atsako kintamojo prognozavimui, kaip ir kiti regresijos modeliai.

2. Praktinė dalis

Šio darbo praktinėje dalyje aukščiau aprašyti kryptinės statistikos modeliavimo metodai bus pritaikyti sumodeliuotiems bei realiems daugiamačiams kryptiniams duomenims. Iš pradžių yra pateikiamas duomenų generavimo ant sferos ir cilindro aprašymas. Šie duomenys yra naudojami, norint pavaizduoti skirtingus modeliavimo metodus bei palyginti juos tarpusavyje. Taip pat yra aprašomi realieji duomenų rinkiniai su kryptiniais kintamaisiais. Toliau yra pateikta sferinių duomenų analizė: sferinis klasterizavimas ir klasifikavimas bei palyginimas su klasikiniiais metodais. Ta pati analizė yra atliekama ir su cilindriniais duomenimis. Galiausiai yra pritaikomi kryptiniai sprendimų medžiai hipercilindriniais duomenims.

Visi skaičiavimai buvo atlikti naudojant R programą (R Core Team, 2019). Kryptinės statistikos skaičiavimams yra naudojami paketai `cicular` (Lund & Agostinelli, 2017), `Directional` (Tsagris *et al.*, 2020) ir `circtree` (Lang *et al.*, 2020b), o grafikams bražyti – paketai `scatterplot3d`, `rgl`, `plot3D` (Ligges & Maechler, 2003; Adler & Murdoch, 2019; Soetaert, 2017). Taip pat, skaičiavimams yra naudojamos naujai sukurtos funkcijos, kurias galima rasti priede A.

2.1. Duomenys

2.1.1. Sumodeliuoti duomenys

Norint pavaizduoti teorinėje dalyje aprašytų kryptinių duomenų modeliavimo metodų veikimą, yra generuojami fiktyvūs duomenys, kurie atitinka dominančius duomenų tipus. Taip pat, duomenų modeliavimas yra naudojamas, skirtingų modeliavimo metodų veikimo – tikslumo bei greičio, tarpusavyje palyginimui.

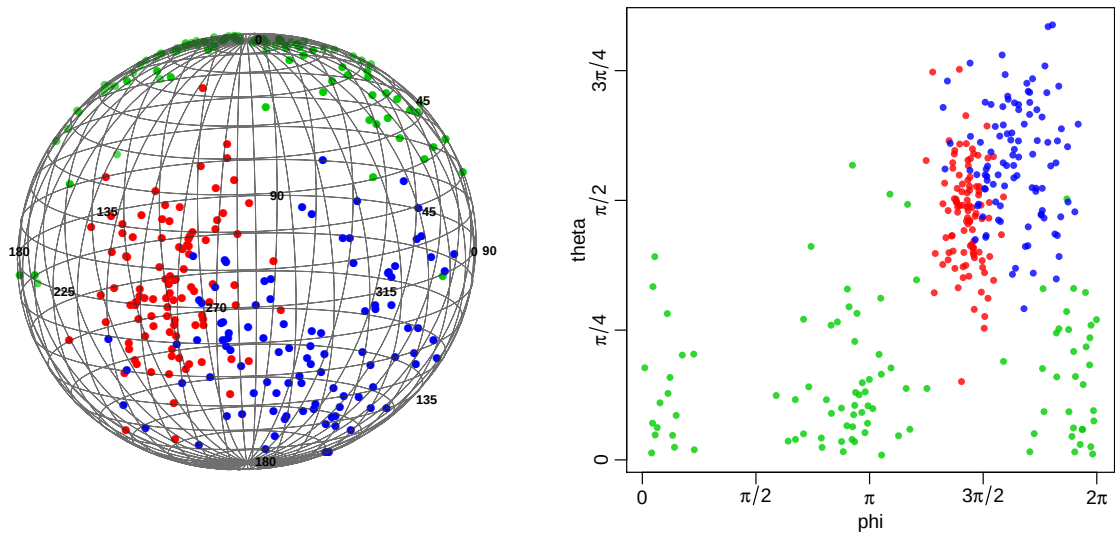
Toliau yra pateikiamas dviejų kryptinių duomenų tipų: sferinio ir cilindrinio, duomenų generavimo metodika.

Sferinių duomenų generavimas. Generuojame atsitiktinius dydžius (θ_{ij}, ϕ_{ij}) iš 3 skirtingų von Mises pasiskirstymų. Kiekvienos atsitiktinių dydžių grupės imties dydis yra 100. Vidutinės kryptys ir koncentracijos parametrai kiekvienai grupei yra nurodyti žemiau (1 lentelė).

1 lentelė. Sugeneruotų sferinių duomenų pasiskirstymų parametrai

N	μ_1	κ_1	μ_2	κ_2
100	1.5	10	4.5	25
100	0	3	3	5
100	1.8	8	5.2	8

Pavaizduojame gautus duomenis ant sferos bei plokštumoje.



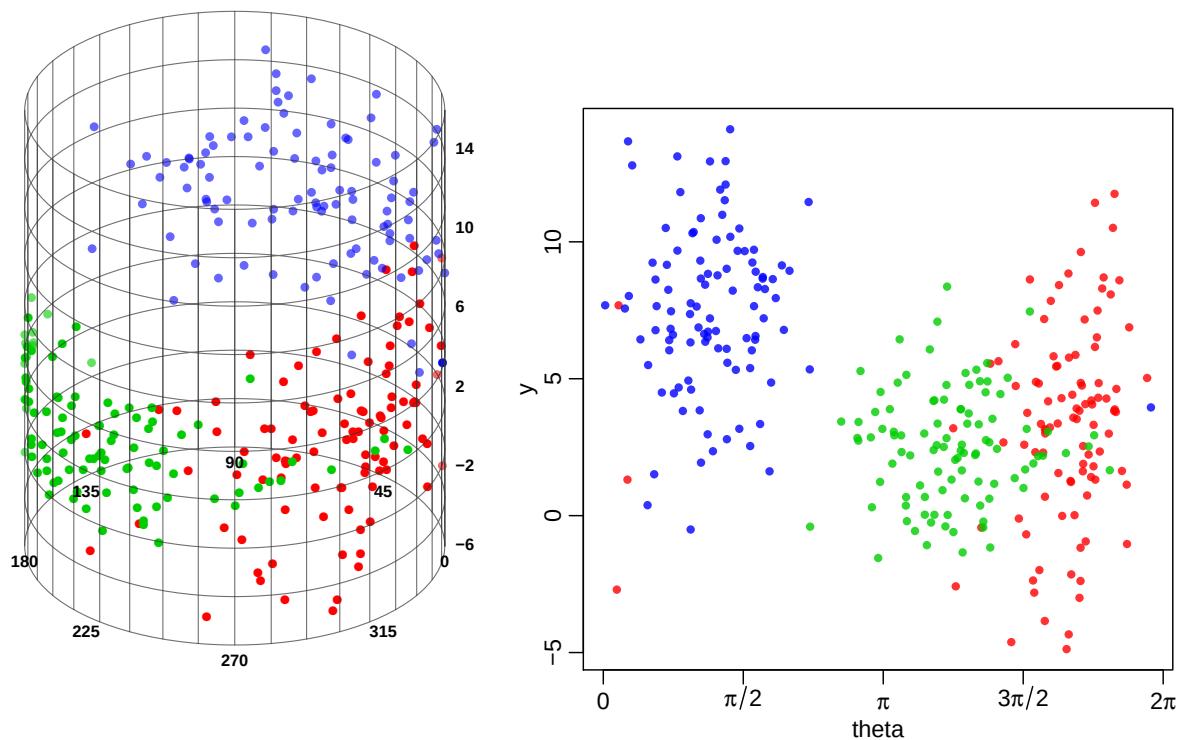
12 pav. Sugeneruoti sferiniai duomenys su 3 grupėmis

Cilindrinų duomenų generavimas. Generuojame atsitiktinius dydžius θ_{ij} iš 3 skirtingų von Mises pasiskirstymų ir atsitiktinius dydžius y_{ij} iš 3 skirtingų normaliųjų (Gau-so) pasiskirstymų. Kiekvienos atsitiktinių dydžių grupės imties dydis yra 100. Vidutinės kryptys, koncentracijos parametrai, vidurkiai ir standartiniai nuokrypiai kiekvienai grupei yra nurodyti žemiau.

2 lentelė. Sugeneruotų cilindrinų duomenų pasiskirstymų parametrai

N	μ_1	κ	μ_2	σ
100	5.3	5	3	4
100	3.9	3	2.5	2
100	1.2	4	7.5	3

Pavaizduojame gautus duomenis ant cilindro bei plokštumoje.



13 pav. Sugeneruoti cilindriniai duomenys su 3 grupėmis

2.1.2. Realus duomenys

Šiame skyrelyje yra aprašyti du realių duomenų rinkiniai: žvaigždžių duomenys ir meteorologiniai duomenys. Šie duomenys atitinkamai turi sferinį ir hipercilindrinį duomenų tipus.

Žvaigždžių duomenų rinkinys. Tai duomenų rinkinys, kuriame yra surinkta informacija apie ryškias žvaigždes (Hoffleit & Warren, 1991). Ryškiųjų žvaigždžių katalogas yra plačiai naudojamas kaip pagrindinių astronominių ir astrofizinių duomenų šaltinis žvaigždėms, kurių ryškumo dydis yra didesnis nei 6,5 balo. Šiame kataloge yra 9110 objektų, iš kurių 9096 yra žvaigždės (14 objektų, yra naujosios arba ekstragalaktiniai objektai, kurie buvo išsaugoti, tačiau dauguma jų duomenų praleista).

Analizės tikslais iš duomenų rinkinio buvo atrinkti sekantys pagrindiniai kintamieji: žvaigždės padėtys dangaus sferoje (ilguma ir platumas) ir grupavimo kintamasis, kuris nurodo sudėtinių žvaigždžių (*angl. multiple-star*) kodus. Kaip papildomi kintamieji, kuriuos galima naudoti klasifikavimui, buvo atrinkti dar 13 kintamųjų, aprašančias žvaigždes:

- *IRflag* – I, jei infraraudonųjų spindulių šaltinis;
- *RAh*, *RAm*, *RA*s – valandos, minutės ir sekundės RA sistemoje J2000, epocha 2000;
- *DE*, *DEd*, *DEm*, *DE*s – ženklas, laipsniai, minutės sekundės Dec sistemoje J2000, epocha 2000;
- *Vmag* – vaizdinis dydis;

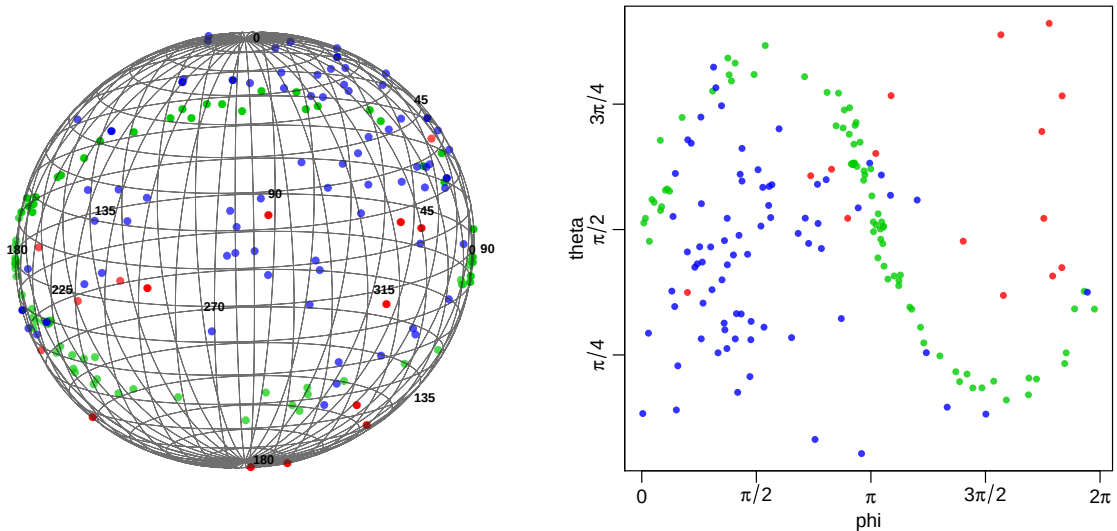
- BV – B-V spalvos UBV sistemoje;
- $pmRA$, $pmDE$ – kasmetinis judėjimas (RA arba Dec) J2000, FK5 sistemoje;
- $RadVel$ – heliocentrinis radialinis greitis.

Taip pat iš duomenų rinkinio buvo išrinktos tik tos žvaigždės, kurioms buvo nurodytas grupavimo kodas, t. y. žvaigždės, kurios yra dvigubos arba sudėtinės. Kodai atitinkantys labai mažą kiekį stebinių buvo sugrupuoti į vieną. Galiausiai duomenų rinkinyje išliko 170 stebinių, kurie yra suskirstyti į tris grupes:

3 lentelė. Sudėtinių žvaigždžių apjungti grupavimo kodai ir jų dažniai

AIR	D	S
15	81	74

Pavaizduojame turimus sferinius duomenis bei jų grupavimą.



14 pav. Sudėtinių žvaigždžių suskirstytų pagal jų kodus padėtis dangaus sferoje

Meteorologinių duomenų rinkinys. Šiame duomenų rinkinyje yra sukaupti istoriniai meteorologiniai duomenys Kanadoje, tiksliau Ontario mieste. Duomenys galima rasti internetiniame puslapyje <https://climate.weather.gc.ca/>. Analizės tikslais iš duomenų bazės buvo parsųsti paskutinių metų duomenys, kurie buvo renkami kas valandą. Taip pat iš turimo duomenų rinkinio buvo išrinkti kintamieji, kurie turi reikšmes beveik visiems stebiniams, ir buvo pašalinti stebiniai, kurie turi praleistų reikšmių pasirinktuose kintamuosiuose. Taigi galiausiai duomenų rinkinys turi 8951 stebinių ir 8 kintamuosius. Šie kintamieji yra:

- $Date$ – data, kada buvo daromi matavimai;

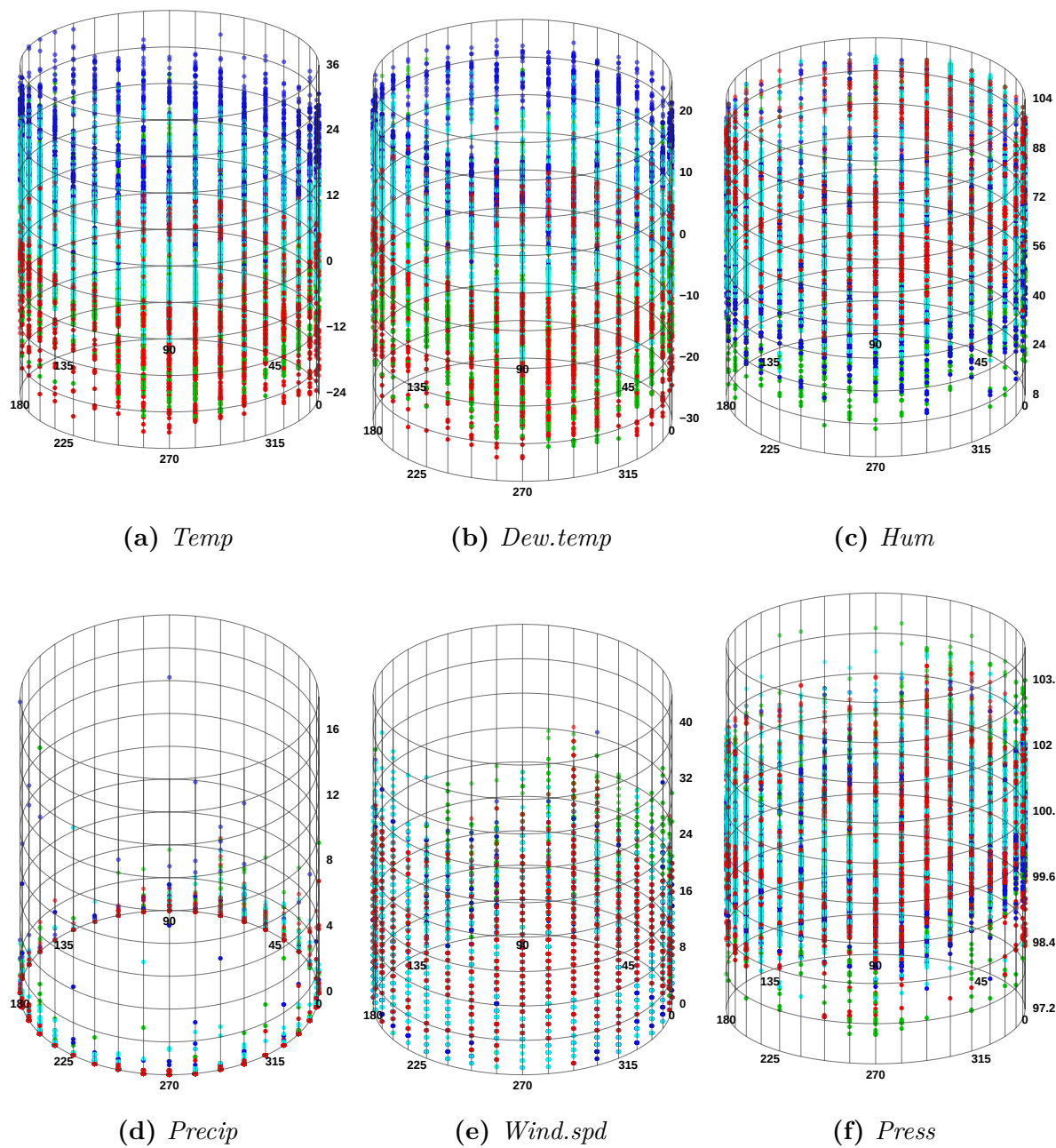
- *Temp* – oro temperatūra;
- *Dew.temp* – rasos temperatūra;
- *Hum* – santykinė drėgmė;
- *Precip* – kritulių kiekis;
- *Wind.dir* – vėjo kryptis;
- *Wind.spd* – vėjo greitis;
- *Press* – slėgis.

Kaip grupavimo kintamąjį pasirenkame metų laikus. Tokio kintamojo dažniai yra pateikti lentelėje žemiau.

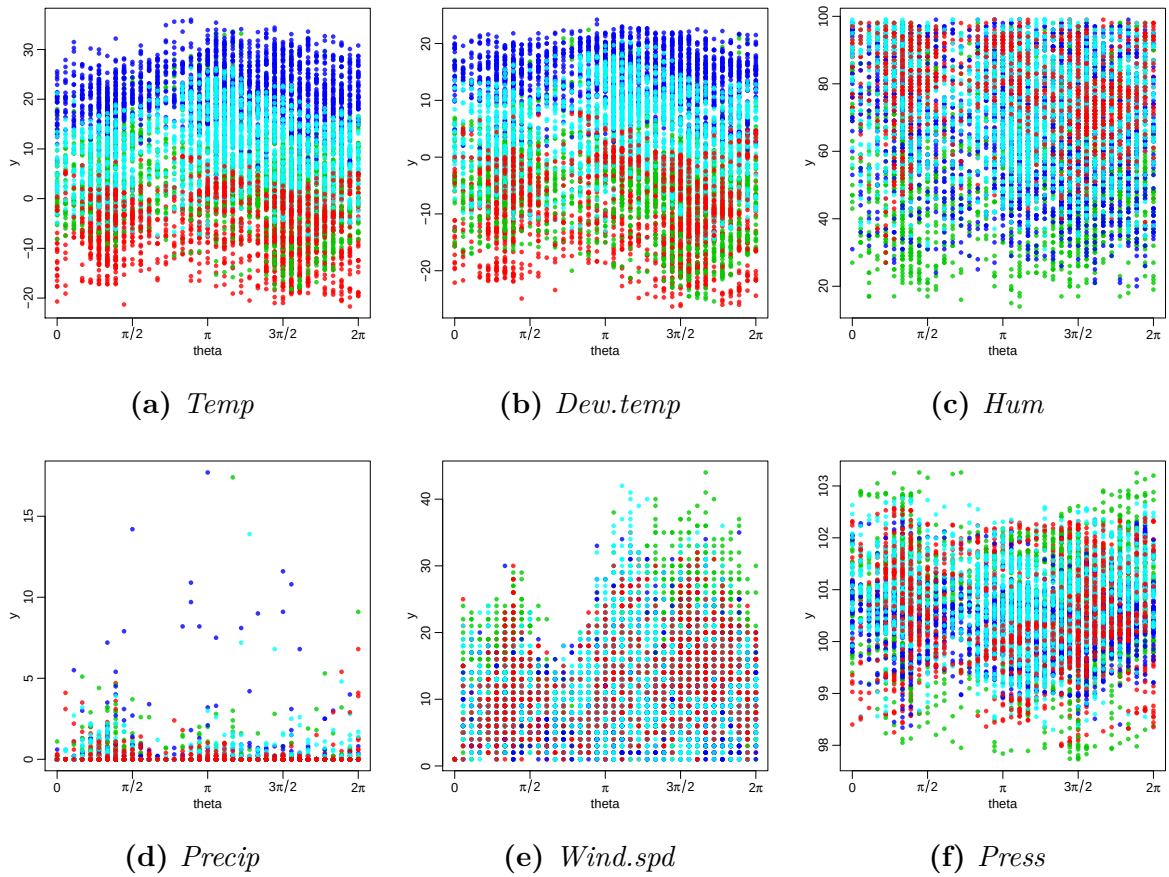
4 lentelė. Meteorologinių duomenų metų laikai ir jų dažniai

Win	Spr	Sum	Aut
2094	2624	2124	2109

Šiame duomenų rinkinyje skritulinis kintamasis yra vėjo kryptis, o visi likusieji kintamieji yra tiesiniai, t. y. kiekvienas kintamasis kartu su vėjo kryptimi sudaro cilindrinį kintamąjį. Pavaizduojame tokių cilindrinį duomenų pasiskirstymus bei grupes ant cilindro bei plokštumoje.



15 pav. Meteorologinių duomenų grupavimo pagal metų laikus pasiskirstymai ant cilindro



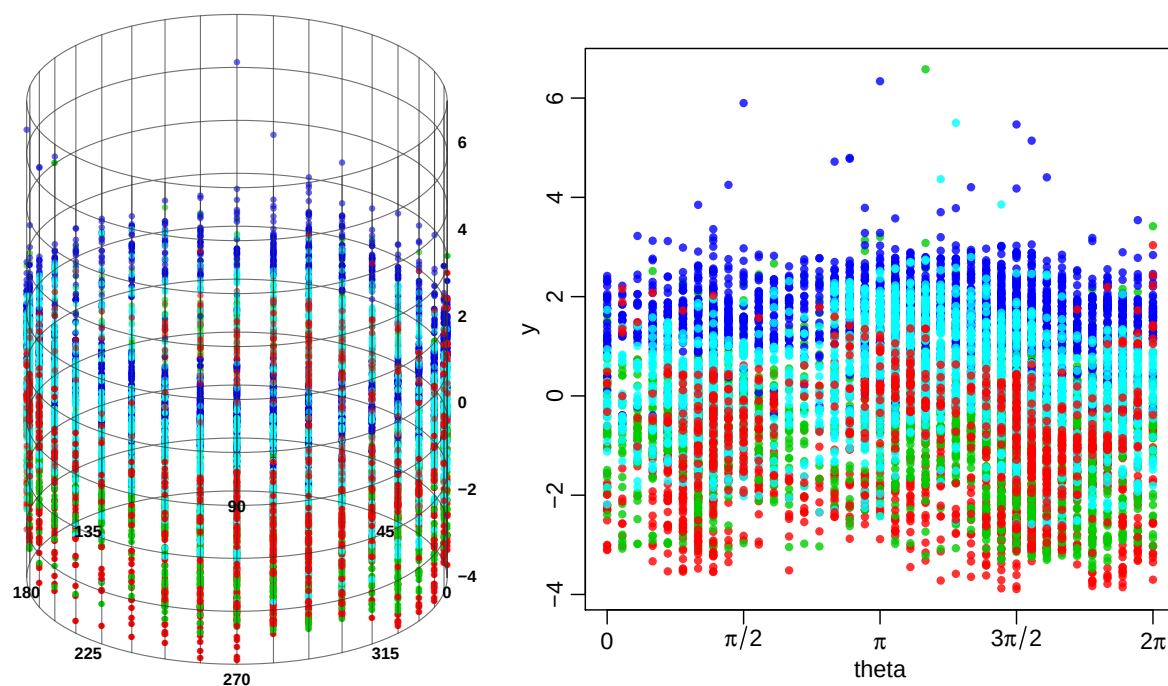
16 pav. Meteorologinių duomenų grupavimo pagal metų laikus pasiskirstymai

Kadangi tokius duomenis vaizduoti nėra patogu, sumažiname tiesinių kintamųjų dimensiją naudojant pagrindinių komponentių metodą. Atvaizdavimo tikslais naudosime tik tai pirmą pagrindinę komponentę. Jos sukaupta informacija sudaro 36% ir jos koeficientai yra:

5 lentelė. Pirmos pagrindinės komponentės iš meteorologinių tiesinių kintamųjų koeficientai

Temp	Dew.temp	Hum	Precip	Wind.spd	Press
0,595	0,65	0,16	0,153	-0,0355	-0,417

Pavaizduojame duomenų grupių pasiskirstymus ant sumažintos dimensijos cilindro bei plokštumoje.



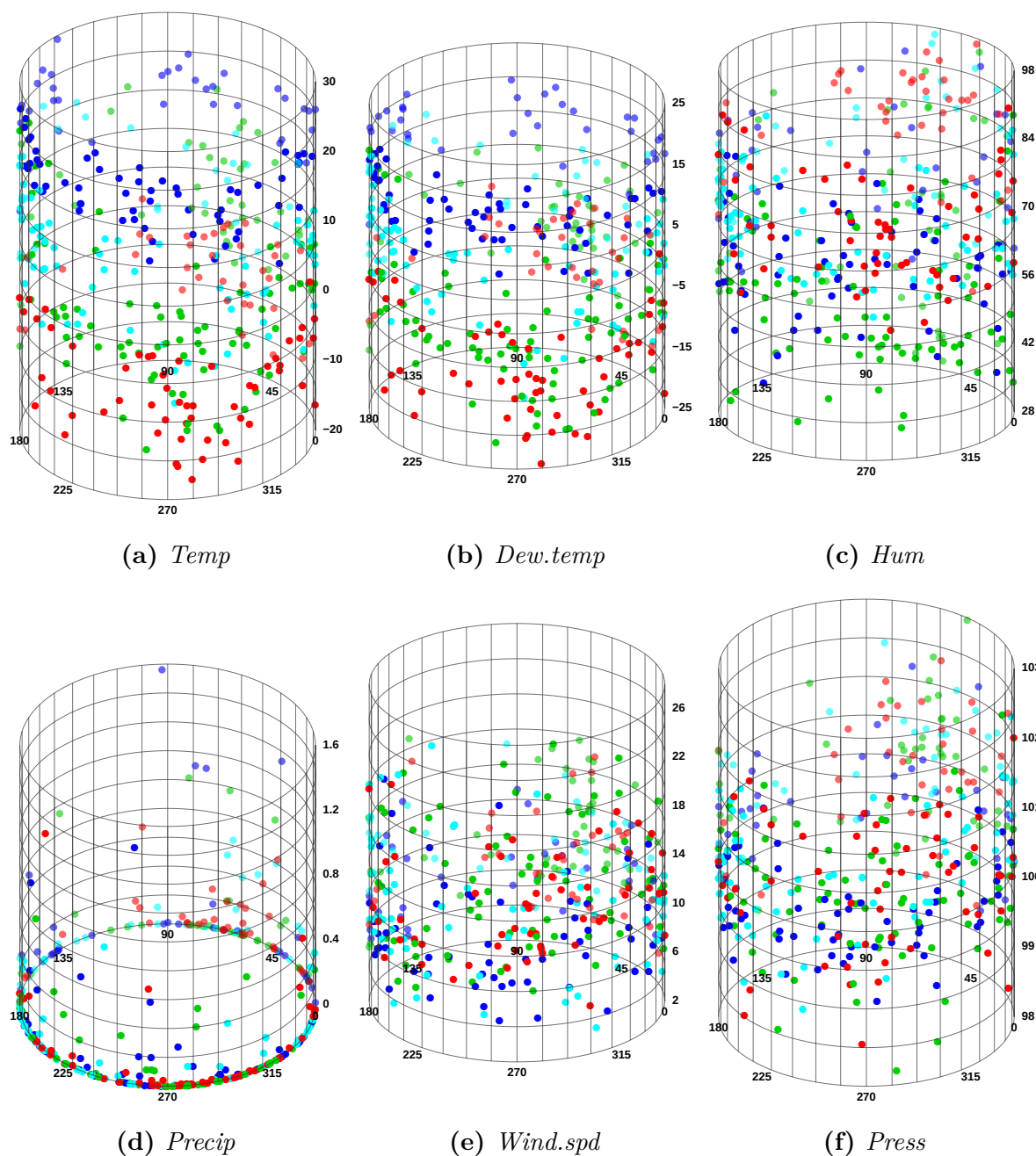
17 pav. Meteorologinių duomenų grupavimas pagal metų laikus ant sumažintos dimensijos cilindrinio duomenų

Analizės bei rezultatų atvaizdavimo tikslais toks duomenų rinkinys gali būti per didelis, todėl iš turimo duomenų rinkinio pasidarome dieninių meteorologinių duomenų rinkinį. Naujas rinkinys turi 385 stebinių. Metų laikų dažniai dabar yra sekantys:

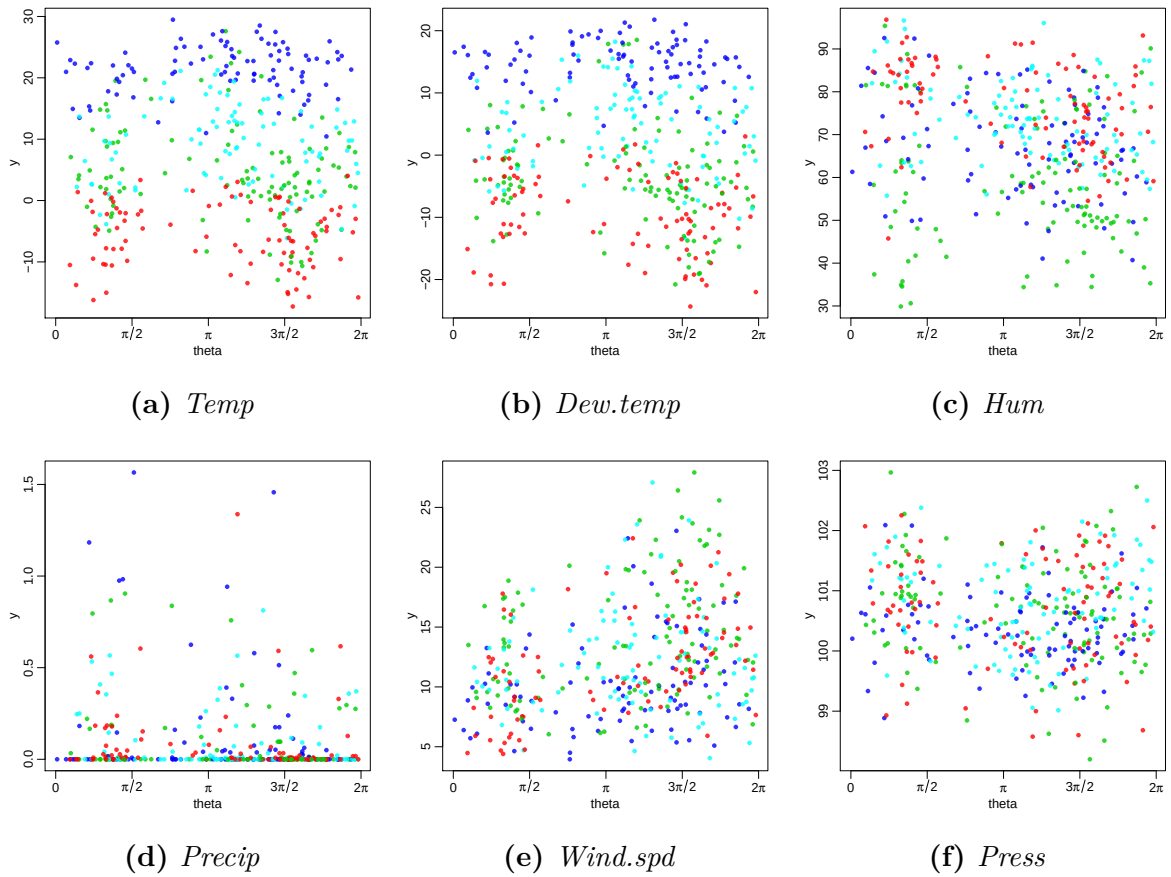
6 lentelė. Dieninių meteorologinių duomenų metų laikai ir jų dažniai

Win	Spr	Sum	Aut
90	112	92	91

Žemiau yra pavaizduoti duomenų pasiskirstymai bei grupės ant cilindro bei plokštumoje.



18 pav. Dieninių meteorologinių duomenų grupavimo pagal metų laikus pasiskirstymai ant cilindro



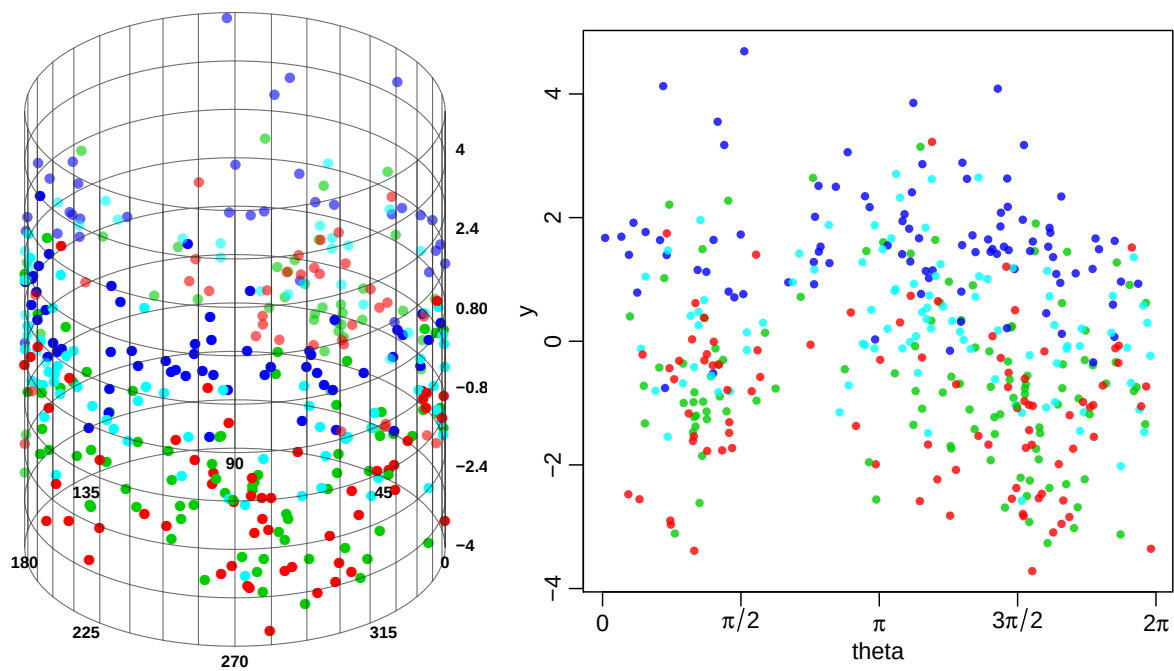
19 pav. Dieninių meteorologinių duomenų grupavimo pagal metų laikus pasiskirstymai

Kaip ir pradiniams valandiniais duomenimis, įvertiname 1 pagrindinę komponentę iš tiesinių kintamųjų. Jos sukaupta informacija sudaro 39% ir jos koeficientai yra:

7 lentelė. Pirmos pagrindinės komponentės iš meteorologinių tiesinių kintamųjų koeficientai

Temp	Dew.temp	Hum	Precip	Wind.spd	Press
0,522	0,593	0,282	0,34	-0,0551	-0,422

Pavaizduojame duomenų grupių pasiskirstymus ant sumažintos dimensijos cilindro bei plokštumoje.



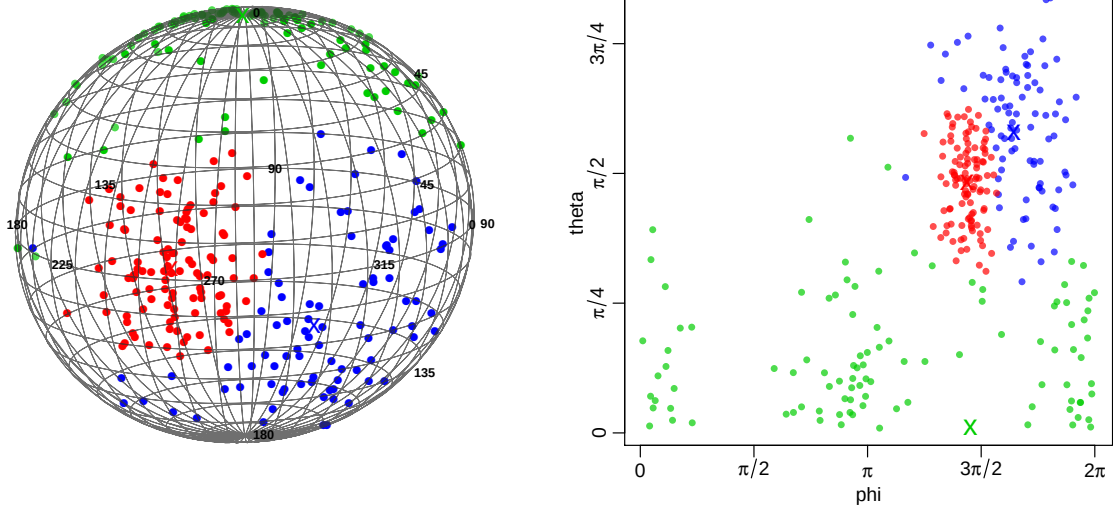
20 pav. Dieninių meteorologinių duomenų grupavimas pagal metų laikus ant sumažintos dimensijos cilindrinės duomenų

2.2. Sferinių duomenų klasterizavimas

Šiame skyrelyje yra pritaikomi du sferinio klasterizavimo metodai: EM klasterizavimas ir k-vidurkių metodas, sumodeliuotiems sferiniams duomenims. Šių algoritmų veikimas yra lyginamas su klasikiniu k-vidurkių klasterizavimu.

2.2.1. EM klasterizavimas

Pritaikome modifikuotą sferiniams duomenims EM klasterizavimo metodą aukščiau sugeneruotiems duomenims. Grupavimo rezultatus pavaizduojame grafiškai ant sferos bei plokštumoje.



21 pav. EM klasterizavimas pritaikytas von Mises-Fisher pasiskirstymo duomenims

Šiam metodui pasiekti konvergavimą užtruko 21 iteraciją. Iš grafiko matome, kad trys grupės atskiriamos pakankamai gerai.

Žemiau yra pateikiami von Mises-Fisher parametrų įverčiai, taip yra įvertinti kiekvienos grupės dydžiai ir EM algoritmo priklausimo klasėms tikimybės.

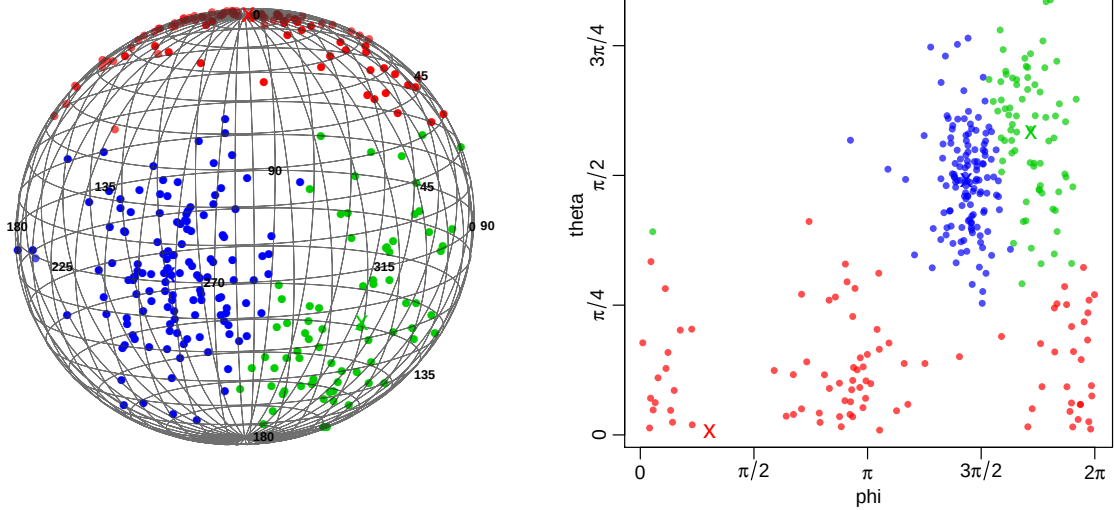
8 lentelė. Sferinio EM klasterizavimo rezultatai: klasterių dydžiai, von Mises-Fisher pasiskirstymo parametrai ir tikimybės (kairėje), tikrų grupių pasiskirstymo parametrai iš lentelės 1 (dešinėje)

N	$\bar{\theta}$	$\bar{\phi}$	κ	p	N	μ_1	κ_1	μ_2	κ_2
110	1.517	4.519	19.265	0.367	100	1.5	10	4.5	25
102	0.033	4.563	6.057	0.340	100	0	3	3	5
88	1.820	5.161	5.464	0.293	100	1.8	8	5.2	8

Matome, kad klasterių dydžiai yra apie 100 ir atitinką sumodeliuotų duomenų grupių dydžius. Priklausymo klasėms tikimybės yra apie 0,3, kas taip pat atitinka tikrovę. Palyginus gautus parametrų įverčius su pasirinktais modeliavimui, matome, kad pirmo ir trečio klasterio vidutinės kryptys sutampa, o antro truputį skiriasi, bet vis tiek yra panašios. Vadinasi sferinis EM klasterizavimo būdas yra geras metodas kryptinių duomenų grupavimui.

2.2.2. Klasterizavimas naudojant k-vidurkių metodą

Toliau pritaikome sferinį k-vidurkių klasterizavimo metodą aukščiau sugeneruotiems duomenims. Grupavimo rezultatus pavaizduojame grafiškai ant sferos bei plokštumoje.



22 pav. Sferinis k-vidurkių klasterizavimas

Šis metodas iki konvergavimo užtruko 6 iteracijas, kas yra daug greičiau nei sferinis EM algoritmas. Jis taip pat, kaip ir EM metodas, sugebėjo išskirti 3 grupes. Beje, šie du metodai duoda panašų stebinių grupavimą.

Įvertiname kiekvienos grupės dydį ir vidutinę kryptį.

9 lentelė. Sferinio k-vidurkių metodo klasterizavimo rezultatai: klasterių dydžiai ir vidutinės kryptys (kairėje), tikrų grupių pasiskirstymo parametrai iš lentelės 1 (dešinėje)

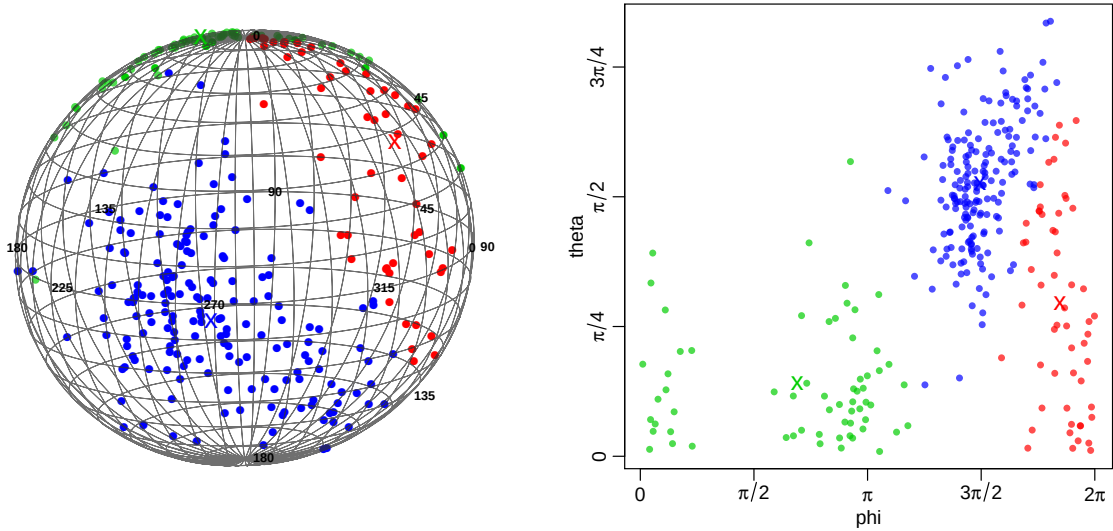
N	$\bar{\theta}$	$\bar{\phi}$	N	μ_1	κ_1	μ_2	κ_2
93	0.023	0.959	100	1.5	10	4.5	25
74	1.830	5.400	100	0	3	3	5
133	1.540	4.508	100	1.8	8	5.2	8

Matome, kad klasterių dydžiai yra arti 100, kas atitinką sumodeliuotų duomenų grupių dydžius, tačiau grupių dydžiai labiau skiriasi nuo tikrovės, nei EM algoritmo. Palyginus gautus parametrų įverčius su pasirinktais modeliavimui, matome, kad pirmo klasterio vidutinės kryptys sutampa, o antro bei trečio truputi skiriasi, bet ne žymiai. Vadinasi sferinis k-vidurkių klasterizavimo būdas yra geras metodas kryptinių duomenų grupavimui.

Palyginus sferinį EM ir k-vidurkių klasterizavimus, gauname, kad šiems sugeneruotiems duomenims k-vidurkių algoritmas yra daug greitesnis, tačiau EM metodas duoda tikslesnius rezultatus.

2.2.3. Palyginimas su klasikiniu klasterizavimu

Toliu palyginimui pritaikysime klasikinį k-vidurkių metodą sumodeliuotiems sferiniams duomenims. Klasterių skaičius, kaip ir anksčiau, yra pasirenkamas lygus 3. Gautus rezultatus vaizduojame grafiškai.



23 pav. Tiesinis k-vidurkių klasterizavimas

Klasikinis k-vidurkių metodas iki konvergavimo užtruko tik 3 iteracijas, kas yra greičiau nei kiti algoritmai. Tačiau matome, kad šis metodas suskirstė taškus į grupes visiškai kitaip. Taip yra dėl to, kad tiesinis algoritmas neatsižvelgia į sferinių duomenų ypatumus, t. y. nežino, kad kintamieji yra periodiniai ir neturi pradžios bei galo taškų.

Įvertiname, kiek stebinių pateko į kiekvieną klasterį. Taip pat pateikiame klasterių centrus.

10 lentelė. Klasikinio k-vidurkių metodo klasterizavimo rezultatai: klasterių dydžiai ir centrai (kairėje), tikrų grupių pasiskirstymo parametrai iš lentelės 1 (dešinėje)

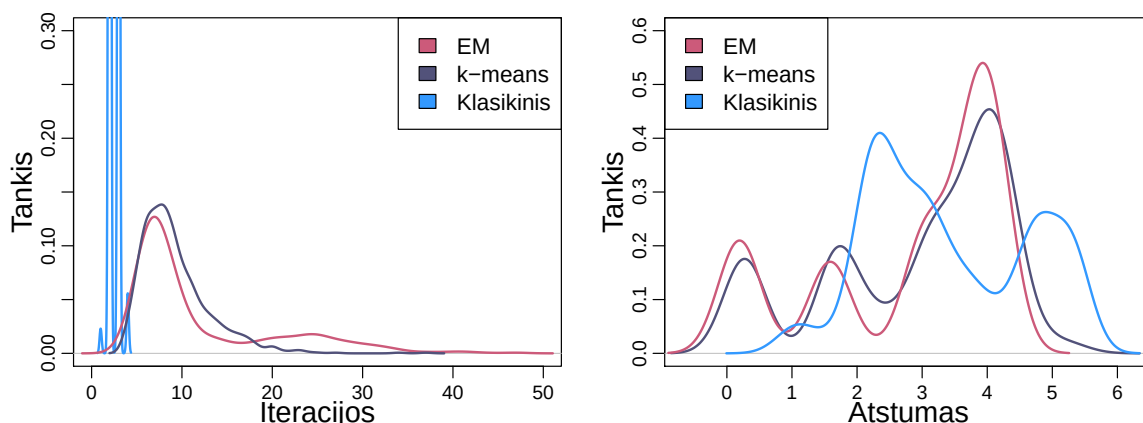
N	$\bar{\theta}$	$\bar{\phi}$	N	μ_1	κ_1	μ_2	κ_2
56	0.925	5.802	100	1.5	10	4.5	25
63	0.439	2.170	100	0	3	3	5
181	1.652	4.697	100	1.8	8	5.2	8

Palyginame, gautus rezultatus su pradiniais grupių parametrais. Matome, kad grupių dydžiai visiškai neatitinka tikrovės. Centrų įverčiai $\bar{\theta}$ ir $\bar{\phi}$ irgi nėra panašūs į parametrų μ_1 ir μ_2 reikšmes. Todėl galima tarti, kad klasikinių klasterizavimo metodų taikymas kryptiniams duomenims – tokiems kaip yra sferiniai duomenys, yra klaidingas. Šis pavyzdys parodo, kodėl prieš taikant klasikinius algoritmus kryptiniams duomenims, juos būtina modifikuoti, kad jie atsižvelgtų į tokių duomenų ypatumą.

2.2.4. Metodų tarpusavyje palyginimas

Šitame skyrelyje skirtingi klasterizavimo ant sferos metodai yra palyginami tarpusavyje. Tam yra naudojamos duomenų pakartotinis modeliavimas ir jų klasterizavimas. Sferiniai duomenys yra modeliuojami, taip pat kaip buvo aprašyta skyriuje 2.1.1, 1000 kartų. Kiekvienu atveju yra pritaikomi trys klasterizavimo metodai: sferinis EM, k-vidurkių ir klasikinis k-vidurkių.

Padarius tokias simuliacijas, yra įvertinami gautų klasterių centrų sferiniai atstumai iki originalių grupių vidutinių krypčių. Jie yra pateikti tankių grafike apačioje. Taip pat yra pateikiami kiekvieno pritaikyto metodo iteracijų iki konvergavimo skaičius.



24 pav. Klasterizavimo ant sferos metodų charakteristikų tankiai: iteracijų skaičius ir sferinis atstumas tarp klasterių centrų ir originalių parametrų

Matome, kad greičiausias algoritmas yra klasikinis k-vidurkių metodas. Sferinis k-vidurkių ir sferinis EM metodai abu turi panašius greičius, tačiau sferinis EM klasterizavimas yra truputi lėtesnis bei jo greitis turi didesnę išsibarstymą. Palyginus nagrinėjamų klasterizavimo metodų tikslumą, matome, kad prasčiausiai veikia klasikinis k-vidurkių metodas, paskui jį pagal blogumą yra sferinis k-vidurkių klasterizavimas, o geriausias yra sferinis EM klasterizavimas. Nors abiejų sferinių klasterizavimo metodų tikslumas skiriasi nedaug.

Įvertiname aukščiau pateiktų charakteristikų vidurkius kiekvienam metodui.

11 lentelė. Klasterizavimo ant sferos metodų vidutinės charakteristikos: iteracijų skaičius ir sferinis atstumas tarp klasterių centrų ir originalių parametrų

	EM	k-vidurkių	Klasikinis
iteracijos	12.109	9.208	2.400
atstumas	2.793	2.922	3.389

Iš lentelės galima padaryti tokias pačias išvadas, kaip ir iš aukščiau pateiktų tankių

grafikų. Vadinasi geriausiai taikyti yra sferinį k-vidurkių klasterizavimo metodą, nes jo tikslumo ir greičio santykis yra optimaliausias.

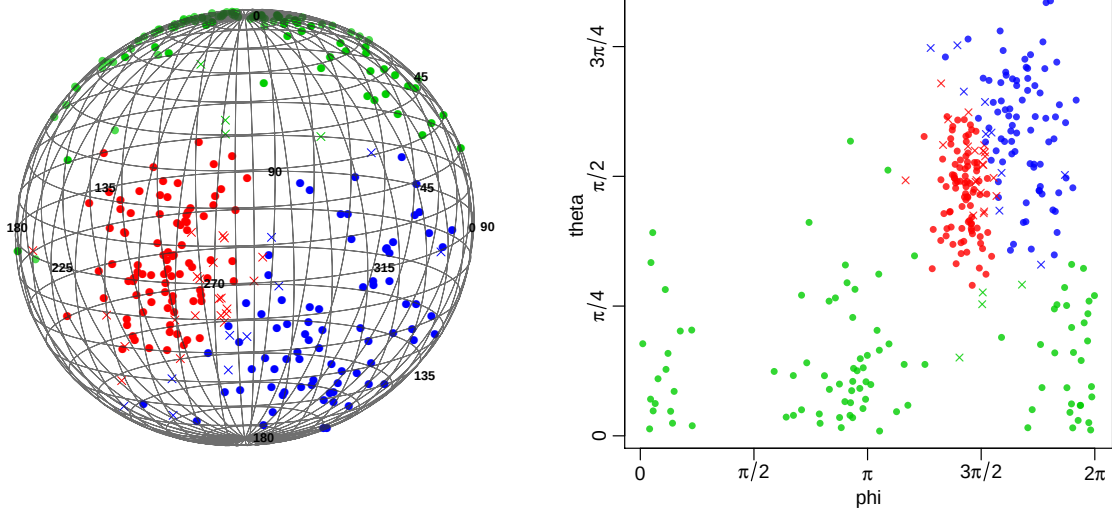
2.3. Sferinių duomenų klasifikavimas

Šiame skyrelyje yra analizuojami tiek sumodeliuoti, tiek realus (žvaigždžių) sferiniai duomenys. Šiems duomenims yra taikomi du sferinio klasifikavimo metodai: diskriminantinė analizė ir logistinė regresija. Žvaigždžių duomenims taip pat yra pritaikomas hibridinės sferinės logistinės regresijos modelis. Visų šių algoritmų veikimas yra lyginamas tarpusavyje bei su klasikine logistine regresija.

2.3.1. Diskriminantinė analizė

Šitame skyrelyje sferinės tiesinės diskriminantinės analizės metodas yra pritaikomas sumodeliuotiems sferiniams duomenims bei realiems žvaigždžių duomenims.

Iš pradžių pritaikome sferinės diskriminantinės analizės metodą sumodeliuotiems duomenims, kurie turi tris klases. Rezultatas yra pavaizduotas žemiau ant sferos bei plokštumoje. Taškiukai žymi stebinius, kurie buvo gerai suklasifikuoti, o kryžiai – stebinius, kurie buvo priskirti netinkamoms klasėms.



25 pav. Sferinės tiesinės diskriminantinės analizės klasifikavimas ant sumodeliuotų duomenų

Taip pat, kad patikrintume kaip gerai veikia toks klasifikavimo algoritmas, įvertiname klasifikavimo tikslumo matricą (*angl. confusion matrix*). Apskaičiuojame kitus skaitinius matus, kurie apibendriną klasifikavimo gerumą:

- teisingai atspėtų dalis arba tikslumas (*angl. accuracy*) – parodo kiek iš viso buvo gerai suklasifikuota;

- specifiškumas (*angl. precision*) – parodo kiek iš visų teigiamų spėjimų buvo gerai suklasifikuoti;
- jautrumas (*angl. recall*) – parodo kiek iš visų teigiamų buvo gerai suklasifikuoti;
- AUC (*angl. Area under curve*) – plotas po ROC kreivę, reikšmė lygi 0.5 reiškia atsitiktinį spėjimą.

Visi šie matai kinta nuo 0 iki 1, geriausia reikšmė yra 1. Kadangi klasių yra 3, tai šie matai yra skaičiuojami kiekvienai klasei atskirai. Taip pat yra pateikiamos šių matų bendro klasifikavimo įverčiai.

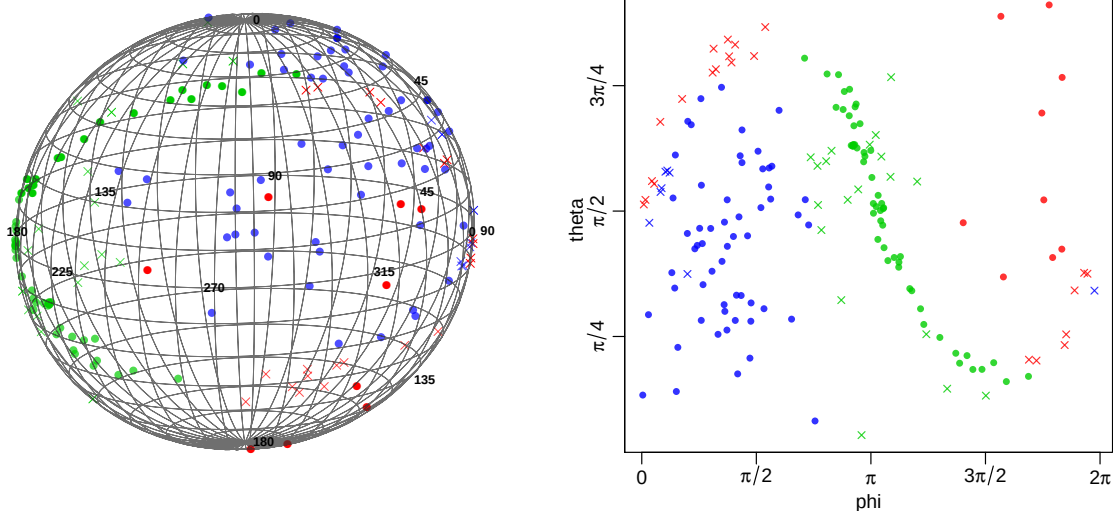
12 lentelė. Sumodeliuotų duomenų sferinės diskriminantinės analizės tikslumo matai

	ACC	PREC	REC	AUC
1	0.893	0.809	0.890	0.892
2	0.977	0.960	0.970	0.975
3	0.897	0.888	0.790	0.870
viso	0.883	0.886	0.883	0.912

Prognozēs				
	1	2	3	
Grupēs	1	89	3	8
	2	1	97	2
	3	20	1	79

Sferinės diskriminantinės analizės metodo tikslumas šiems sumodeliuotiems duomenims yra 0,883, kas reiškia, kad toks algoritmas neblogai gali prognozuoti grupes. Klasifikavimo tikslumo matrica taip pat parodo, kad dauguma stebinių buvo suklasifikuoti teisingai. Labiausiai painiojasi 1 ir 3 grupės, kurios yra išdėstytos šalia viena kitos, todėl tai natūralu.

Toliau pritaikome tą patį sferinės tiesinės diskriminantinės analizės metodą klasifikuojant žvaigždžių duomenis į tris grupes pagal sudėtinių žvaigždžių kodus. Tokio algoritmo klasifikavimo rezultatas yra pateiktas žemiau ant sferos bei plokštumoje. Kaip ir anksčiau taškiukai ir kryžiukai atitinkamai žymi gerai bei blogai suklasifikuotas žvaigždes.



26 pav. Žvaigždžių klasifikavimas naudojant sferinę tiesinę diskriminantinę analizę

Iš grafiko matosi, kad yra pakankamai daug žvaigždžių, kurios yra netinkamai suklasifikuotos (tikros grupės yra 14 pav.). Taip yra dėl to, kad sferinės diskriminantinės analizės metode yra daroma prielaida, kad duomenys yra pasiskirstę pagal von Mises-Fisher pasiskirstymą, o iš grafikų akivaizdų, kad taip nėra.

Kaip ir sumodeliuotų duomenų atveju, įvertiname klasifikavimo klasifikavimo tikslumo matricą, bendrą tikslumą, specifiskumą, jautrumą bei AUC.

13 lentelė. Žvaigždžių sferinės diskriminantinės analizės klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC
AIR	0.735	0.743	0.679	0.733
D	0.835	0.290	0.600	0.729
S	0.853	0.877	0.770	0.843
viso	0.712	0.637	0.683	0.768

Prognozès				
		AIR	D	S
Grupès	AIR	9	5	1
	D	19	55	7
	S	3	14	57

Klasifikavimo tikslumo matrica parodo, kad dauguma stebinių yra gerai suklasifikuota, tačiau yra pakankamai daug atvejų, kai žvaigždėms yra priskiriama netinkama klasė. Bendras sferinės diskriminantinės analizės metodo tikslumas yra 0,712, kas nėra labai blogai.

Palyginus sferinės tiesinės diskriminantinės analizės metodo veikimą duomenų klasifikavimui sumodeliuotiems duomenims ir realiems žvaigždžių duomenims, galima daryti išvadą, kad toks klasifikavimo metodas yra tinkamas tik jei duomenys nors apytiksliai yra pasiskirstę pagal von Mises-Fisher pasiskirstymą, kaip ir yra sumodeliuotų duomenų atveju. Priešingu atveju, kaip nutiko žvaigždžių klasifikavimo atveju, algoritmas nesugebės gerai suklasifikuoti visų duomenų. Taigi, jei duomenys neatitinka sferinės diskriminan-

tinės analizės prielaidos apie pasiskirstymą, norint gauti geras grupių prognozes reikia ieškoti kito klasifikavimo metodo.

2.3.2. Logistinė regresija

Šitame skyrelyje sferinės multinominės logistinės regresijos klasifikavimo modelis, kaip ir diskriminantinė analizė, yra taikomas sumodeliuotiems sferiniams ir žvaigždžių duomenims.

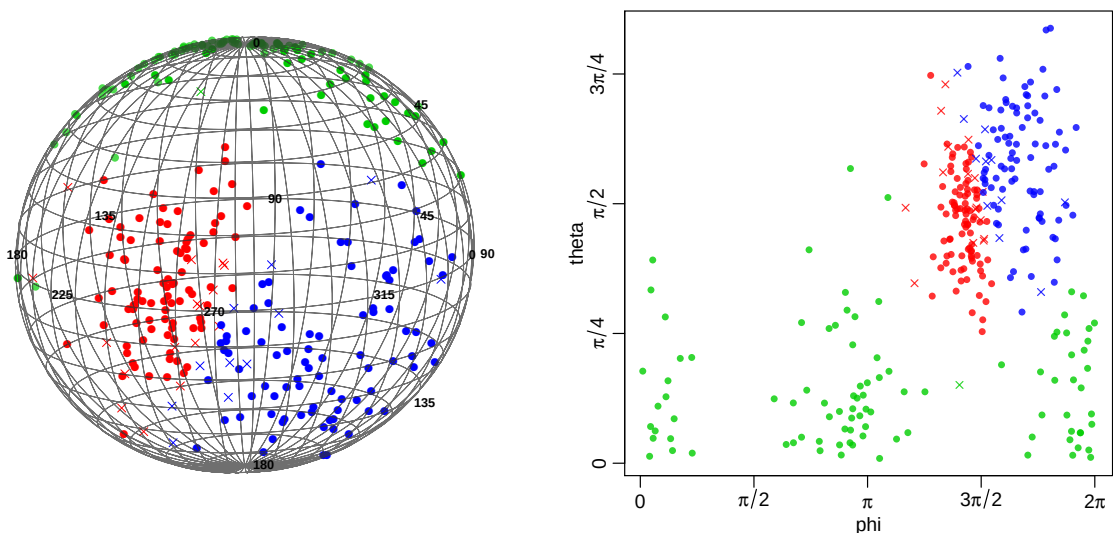
Pritaikius sferinę logistinę regresiją sumodeliuotiems duomenims su trimis grupėmis, gauname modelio tikslumą lygu 0,907. Įvertinta klasifikavimo tikslumo matrica bei kiti klasifikavimo gerumo matai yra pateikti žemiau.

14 lentelė. Sumodeliuotų duomenų sferinės logistinės regresijos klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC	Prognozēs			
1	0.913	0.849	0.900	0.910	Grupēs	1	2	3
2	0.983	0.990	0.960	0.978		90	1	9
3	0.917	0.887	0.860	0.902		2	96	2
viso	0.907	0.908	0.907	0.930	3	14	0	86

Matome, kad beveik visi stebiniai yra suklasifikuoti teisingai. Palyginus su diskriminantinės analizės klasifikavimu (12 lentelė), šis modelis padaro mažiau klaidingų klasių priskyrimų.

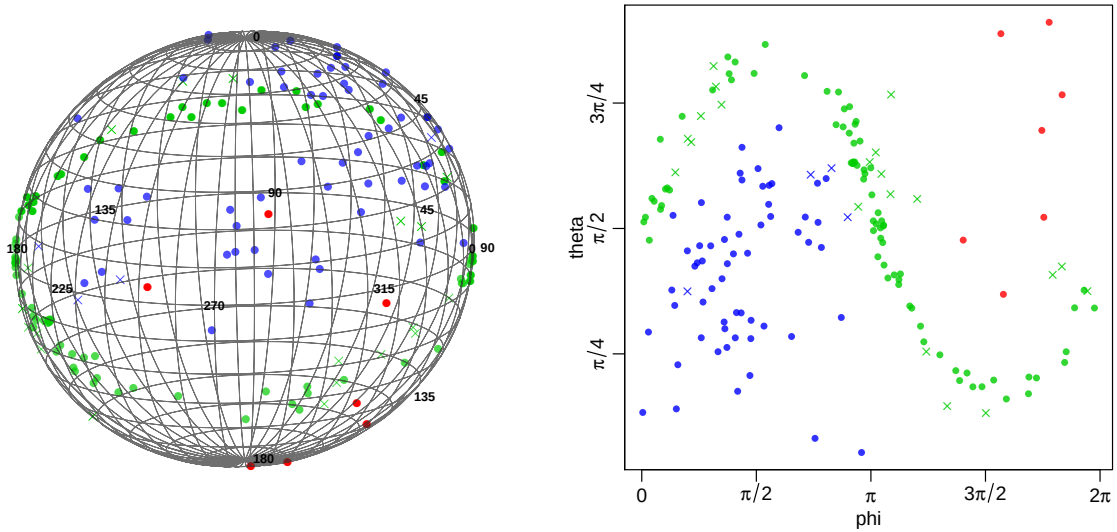
Pavaizduojame šio modelio klasifikavimo rezultatus ant sferos bei plokštumoje.



27 pav. Sferinės logistinės regresijos modelio rezultatas sumodeliuotiems duomenims

Šis grafikas taip pat parodo, kaip sferinės logistinės regresijos modelis sugeba pakankamai gerai suklasifikuoti turimus duomenis.

Toliau pritaikome sferinės logistinės regresijos modelį, norint prognozuoti žvaigždžių grupes. Tokio modelio klasifikavimo rezultatas yra pateiktas žemiau ant sferos bei plokštumoje.



28 pav. Žvaigždžių klasifikavimas naudojant sferinę logistinę regresiją

Matome, kad visos trys grupės yra pakankamai gerai priskiriamos.

Kad įsitikintume sferinės logistinės regresijos modelio gerume, įvertiname klasifikavimo tikslumo matricą ir kitus klasifikavimo gerumo matus: tikslumą, specifiškumą, jautrumą ir AUC.

15 lentelė. Žvaigždžių klasifikavimo su sferine logistine regresija klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC
D	0.882	0.802	1.000	0.888
AIR	0.953	1.000	0.467	0.733
S	0.882	0.935	0.784	0.871
viso	0.859	0.912	0.750	0.831

Prognozès				
		AIR	D	S
Grupès	AIR	7	4	4
	D	0	81	0
	S	0	16	58

Taigi dauguma žvaigždžių yra priskiriama prie tinkamų grupių. Bendras logistinės regresijos tikslumas yra 0,859. Palyginus gautus rezultatus su sferinės diskriminantinės analizės (13 lentelė), matome, kad klaidingų priskyrimų yra daug mažiau. Taip yra dėl to, kad šis modelis yra lankstesnis ir neturiu jokių išankstinių prielaidų apie grupių pasiskirstymus.

Kadangi į sferinės logistinės regresijos modelį galima įtraukti papildomų parametrų, kurie nėra kryptiniai, turima modelį galima bandyti patobulinti įtraukus daugiau duomenų apie žvaigždes, kurie gali būti koreliuoti su žvaigždžių grupėmis. Iš esamų papildomų 13 kintamųjų yra atrenkami kintamieji, kurie pagerina klasifikavimo prognozes. Atrinkimas

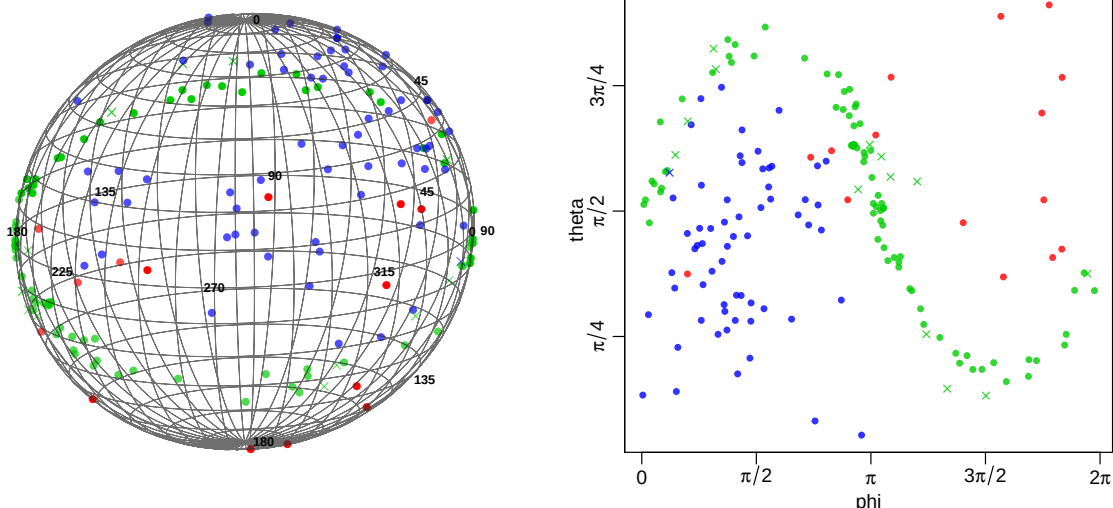
daromas iteraciniu būdu naudojant modelio parametrų reikšmingumo t-statistiką (arba atitinkamą p-reikšmę) ir išmetant po vieną nereikšmingiausi parametras, tol kol modelio visi parametrai bus reikšmingi. Reikšmingumo lygmuo α yra parenkamas lygus 0,05.

Taigi, pritaikius aprašytą būdą, iš 13 kintamųjų buvo atrinkti sekantys: *IRflagI*, *RAs*, *DE*, *DEd*, *Vmag*, *BV*, *pmDE*, *RadVel*. Jų galutinės parametrų reikšmingumo p-reikšmės yra pateiktos lentelėje žemiau.

16 lentelė. Logistinės regresijos modelio su papildomais kintamaisiais parametrų reikšmingumo hipotezės p-reikšmės

	D	S
(Intercept)	1.808e-05	1.251e-05
x	8.246e-07	3.545e-07
y	< 1e-16	< 1e-16
z	< 1e-16	< 1e-16
IRflagI	0.0022	0.0017
RAs	4.273e-13	2.798e-13
DE+	1.161e-07	1.169e-07
DEd	1.805e-13	4.013e-14
Vmag	6.520e-13	1.479e-12
BV	< 1e-16	< 1e-16
pmDE	< 1e-16	< 1e-16
RadVel	9.011e-09	1.329e-08

Pavaizduojame sferinės logistinės regresijos su papildomais kintamaisiais klasifikavimo rezultatai ant sferos bei plokštumoje.



29 pav. Žvaigždžių klasifikavimas naudojant sferinę logistinę regresiją su papildomais kintamaisiais

Iš grafiko, galima padaryti išvada, kad beveik visos žvaigždės yra tinkamai suklasifikuotos.

Įvertiname šio modelio gerumą skaitmeniškai, t. y. apskaičiuojame klasifikavimo tikslumo matricą bei kitus klasifikavimo gerumo matus.

17 lentelė. Žvaigždžių klasifikavimo naudojant sferinę logistinę regresiją su papildomais kintamaisiais klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC	Prognozės		
					AIR	D	S
D	0.882	0.802	1.000	0.888			
AIR	0.953	1.000	0.467	0.733			
S	0.882	0.935	0.784	0.871			
viso	0.859	0.912	0.750	0.831			

Grupės		Prognozės		
		AIR	D	S
	AIR	7	4	4
	D	0	81	0
	S	0	16	58

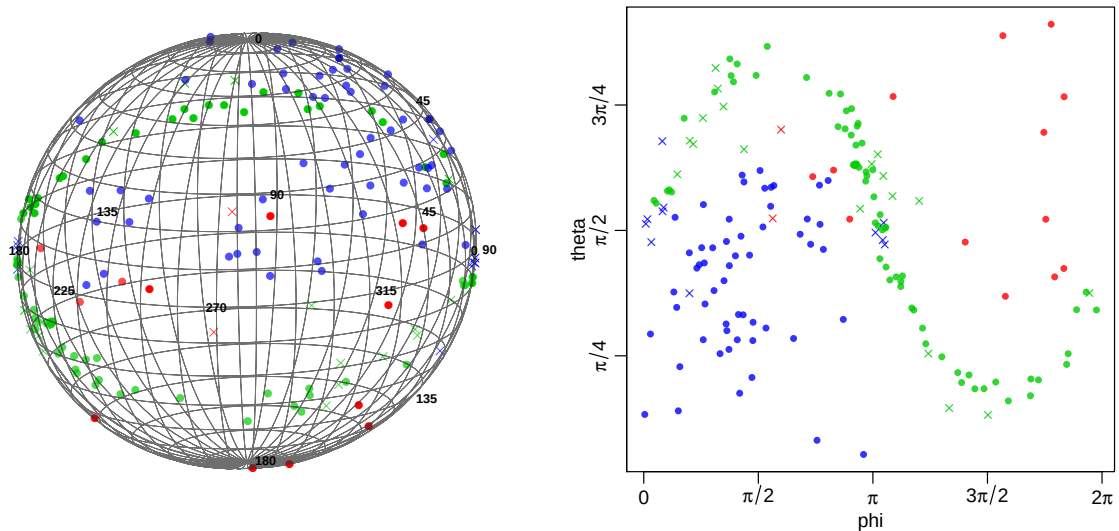
Sferinės logistinės regresijos su papildomais parametrais modelio bendras tikslumas yra 0,918. Klasifikavimo tikslumo matrica taip pat parodo, kad beveik visi stebiniai yra tinkamai priskirti prie klasių. Šis modelis galima sakyti suklysta tik vienoje vietoje – painiojant žvaigždžių grupę **S** su grupe **D**. Palyginus šiuos rezultatus su modelių be papildomų parametru (15 lentelė), modelis su papildomais parametrais klasifikuoja geriau, kaip teoriškai ir turėtų būti.

2.3.3. Palyginimas su klasikiniu klasifikavimu

Šitame skyrelyje yra pritaikomas klasikinis tiesinis klasifikavimo algoritmas sferiniams duomenims. Tai daroma, kad parodytume, kodėl klasikinės statistikos bei modeliavimo

metodai yra netinkami kryptiniams duomenims. Skirtingai nei praeituose skyreliuose, čia yra taikomas tik vienas modelis ir tik realiems žvaigždžių duomenims.

Atsižvelgiant į tai, kad logistinės regresijos modelis yra lankstesnis nei diskriminantinės analizės, nes neturi prielaidų apie pradinius grupių pasiskirstymus, toliau bus taikomas tik logistinės regresijos modelis. Taip pat, kad galėtume palyginti rezultatus su sferiniu atveju, į modelį yra įtraukiami tokie patys, aukščiau išrinkti, papildomi kintamieji. Žemiau yra pavaizduotas tokio klasifikavimo modelio rezultatas ant sferos bei plokštumoje.



30 pav. Žvaigždžių klasifikavimas naudojant klasikinę logistinę regresiją su papildomais kintamaisiais

Iš grafiko matome, kad duomenys yra pakankamai neblogai suklasifikuoti, tačiau sferinis šio modelio analogas veikia geriau.

Kad įsitikintume aukščiau padarytos išvados teisingumu, įvertiname tokio klasifikavimo klasifikavimo tikslumo matricą bei kitus, aukščiau minėtus, 4 klasifikavimo gerumo matavimus.

18 lentelė. Žvaigždžių klasifikavimo naudojant klasikinės logistinės regresijos modelį su papildomais kintamaisiais klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC
D	0.835	0.798	0.877	0.837
AIR	0.976	0.867	0.867	0.927
S	0.824	0.833	0.743	0.814
viso	0.818	0.833	0.829	0.859

Prognozės				
		AIR	D	S
Grupės	AIR	13	1	1
	D	0	71	10
	S	2	17	55

Bendras klasikinės logistinės regresijos modelio tikslumas yra 0,818, kas yra žymiai mažiau nei sferinio analogo atveju gautas tikslumas. Be to, klasifikavimo tikslumo matrica

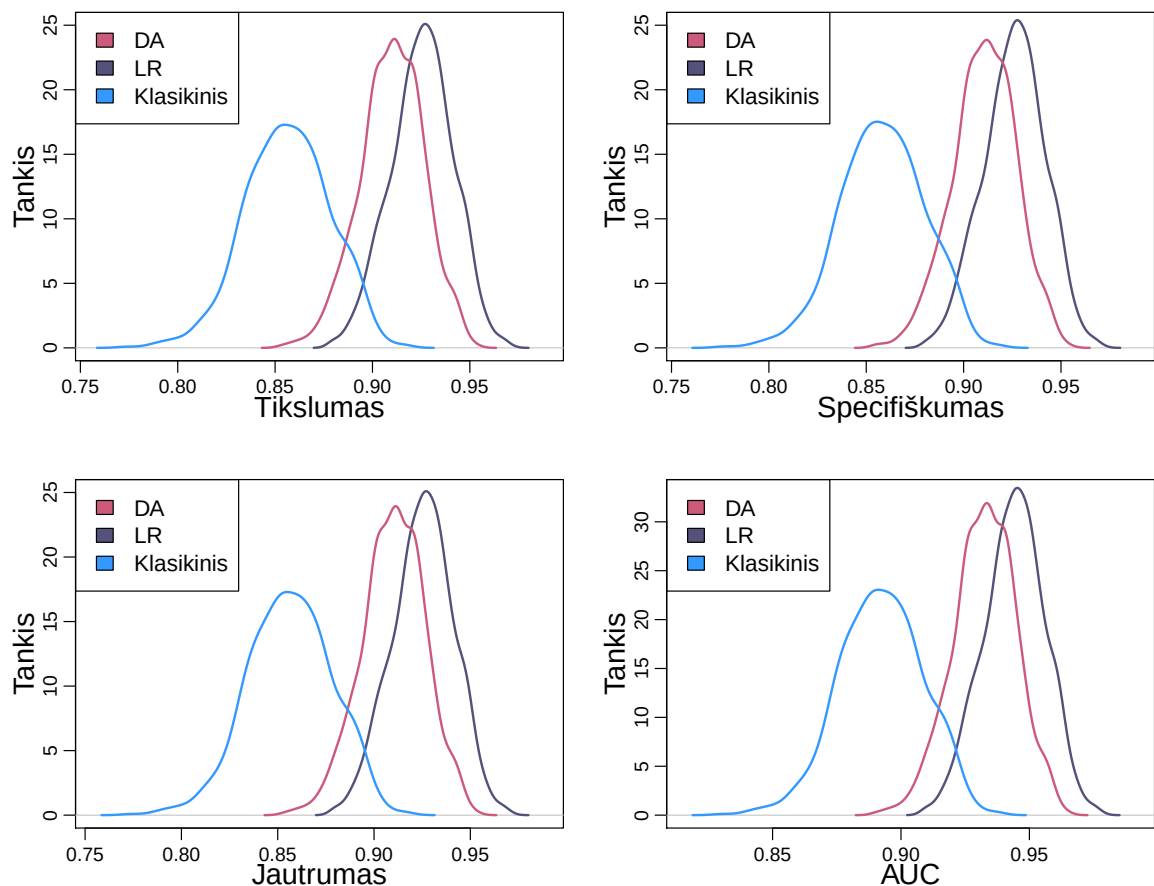
taip pat parodo, kad šiuo atveju yra daroma daugiau klaidingų priskyrimų. Taip pat reikia nepamiršti, kad klasikinio modelio atveju, pateikus tas pačias kryptys tačiau su kitomis skaitmenimis koordinatėmis, galima gauti visiškai skirtingus rezultatus.

Taigi sferinė logistinė regresija yra geresnis sferinių duomenų klasifikavimo būdas nei sferinė diskriminantinė analizė arba klasikinis klasifikavimo būdas.

2.3.4. Metodų tarpusavyje palyginimas

Šitame skyrelyje skirtingi klasifikavimo ant sferos metodai yra palyginami tarpusavyje. Tam yra naudojamos duomenų pakartotinis modeliavimas ir jų klasifikavimas, panašiai kaip klasterizavimo atveju. Sferiniai duomenys yra modeliuojami, taip pat kaip buvo aprašyta skyriuje 2.1.1, 1000 kartu. Kiekvienu atveju yra pritaikomi trys klasifikavimo metodai: sferinė diskriminantinė analizė, sferinė logistinė regresija ir klasikinė logistinė regresija.

Padarius tokias simuliacijas, yra įvertinamas gautų klasių prognozių gerumas. Tam vertinami jau minėti klasifikavimo gerumo matai: tikslumas, specifiškumas, jautrumas ir AUC. Jie yra pateikti tankių grafikuose apačioje.



31 pav. Klasifikavimo ant sferos metodų charakteristikų tankiai: tikslumas, specifiškumas, jautrumas ir AUC matai

Iš tankių matome, kad visi klasifikavimo gerumo matai yra aukščiausi sferinės logistinės regresijos atveju, o žemiausi klasikinės logistinės regresijos atveju.

Įvertiname aukščiau pateiktų charakteristikų vidurkius kiekvienam metodui.

19 lentelė. Klasifikavimo ant sferos metodų vidutinės charakteristikos: tikslumas, specifiškumas, jautrumas ir AUC matai

	ACC	PREC	REC	AUC
DA	0.910	0.911	0.910	0.933
LR	0.925	0.926	0.925	0.944
Klasikinis	0.856	0.858	0.856	0.892

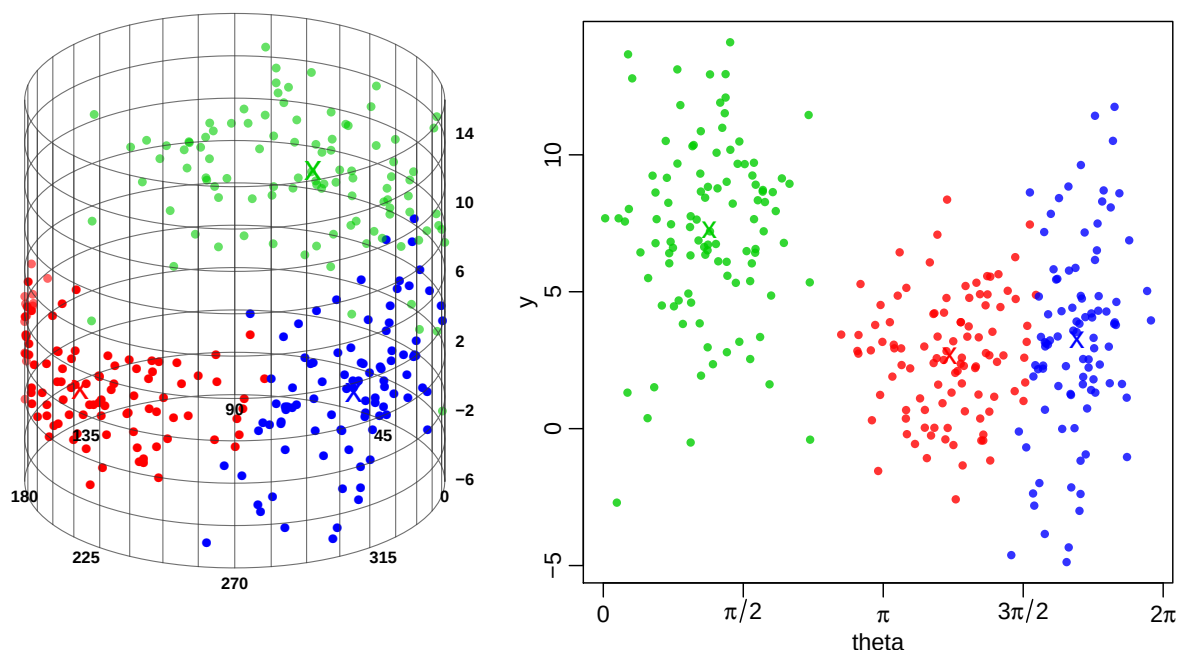
Iš lentelės matome, kad sferiniai klasifikavimo metodai daro tikslesnes prognozes nei klasikinis klasifikavimo metodas. Pats geriausias sferinių duomenų klasifikavimas metodas yra sferinė logistinė regresija.

2.4. Cilindrinų duomenų klasterizavimas

Šiame skyrelyje yra pritaikomi keturi cilindrinio klasterizavimo metodai: EM von Mises-Normalusis (EMvMN) klasterizavimas, SWGMM klasterizavimas, cyk-vidurkių metodas ir fiksuotas cyk-vidurkių metodas, sumodeliuotiems cilindriniam duomenims. Šių algoritmų veikimas yra lyginamas tarpusavyje bei su klasikiniu k-vidurkių klasterizavimu.

2.4.1. EM klasterizavimas

Pritaikome modifikuotą cilindriniam duomenims EM klasterizavimo metodą – von Mises-Normalųjį EM klasterizavimą (EMvMN) aukščiau sugeneruotiems cilindriniam duomenims. Grupavimo rezultatus pavaizduojame grafiškai ant cilindro bei plokštumoje.



32 pav. Cilindrinų duomenų klasterizavimas naudojant EMvMN metodą

Šiam metodui pasiekti konvergavimą užtruko 30 iteracijų. Iš grafiko matome, kad tris grupės atskiriamos pakankamai gerai.

Žemiau yra pateikiami von Mises-Normaliojo pasiskirstymo parametrų įverčiai, taip yra įvertinti kiekvienos grupės dydžiai ir EM algoritmo priklausimo klasėms tikimybės.

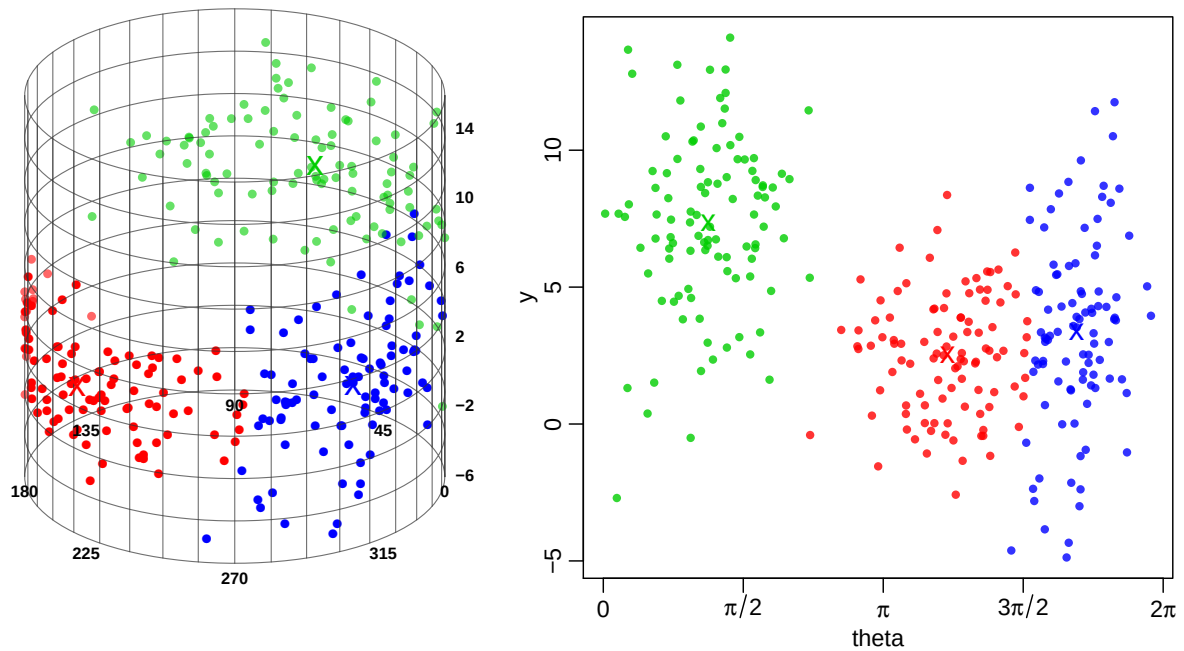
20 lentelė. Suklasterizuotų duomenų von Mises-Normaliojo pasiskirstymo parametrai kiekvienai grupei (kairėje), tikrų grupių modeliavimo parametrai iš lentelės 2 (dešinėje)

	Grupė 1	Grupė 2	Grupė 3					
N	102	103	95	N	μ_1	κ	μ_2	σ
π	0.363	0.343	0.294	100	5.3	5	3	4
μ_0	3.885	1.188	5.316	100	3.9	3	2.5	2
κ	4.236	4.246	9.679	100	1.2	4	7.5	3
μ	5.31	4.513	-1.083					
σ	2.17	3.255	3.536					
ρ	0.223	0.157	0.324					
ν	0.524	1.308	5.974					

Matome, kad klasterių dydžiai yra apie 100 ir atitinką sumodeliuotų duomenų grupių dydžius. Priklausymo klasėms tikimybės yra apie 0,3, kas taip pat atitinka tikrovę. Palyginus gautus parametrų įverčius su pasirinktais modeliavimui, matome, kad skritulinio kintamojo vidutinės kryptis apvalinus sutampa, o apie tiesinio kintamojo vidurkių

sutapimą iš pasiskirstymo parametrų yra sunku spręsti nes jo pasiskirstymas yra sąlyginis.

Toliau pritaikome kitą EM klasterizavimo analogą cilindriniais dydžiais – EM dalinai susukto normaliojo klasterizavimą (EM-SWG), tiems patiems sugeneruotiems cilindriniais duomenims. Šio algoritmo klasterizavimo rezultatas yra pateiktas žemiau ant cilindro bei plokštumoje.



33 pav. Cilindrinų duomenų klasterizavimas naudojant EM-SWG metodą

Šis metodas užtruko 42 iteracijas, kol pasiekia konvergavimą. Pradinių grupių parametrų vertinimui prireikia atitinkamai 1, 15 ir 3 iteracijų. Matome, kad palyginus su EMvMN algoritmu, šis algoritmas yra žymiai lėtesnis. Be to, verta paminėti, kad EM-SWG metodo vienos iteracijos skaičiavimai taip pat užtrunka žymiai ilgiau nei EMvMN metodo. Tačiau iš grafiko matome, kad šis algoritmas taip pat gerai sugebėjo išskirti tris grupes.

Žemiau yra pateikiami SWG pasiskirstymo vidutinių vektorių įverčiai, taip yra įvertinti kiekvienos grupės dydžiai ir EM algoritmo priklausimo klasėms tikimybės.

21 lentelė. Suklasterizuotų duomenų SWG pasiskirstymo vidurkių parametrai kiekvienai grupei (kairėje), tikrų grupių modeliavimo parametrai iš lentelės 2 (dešinėje)

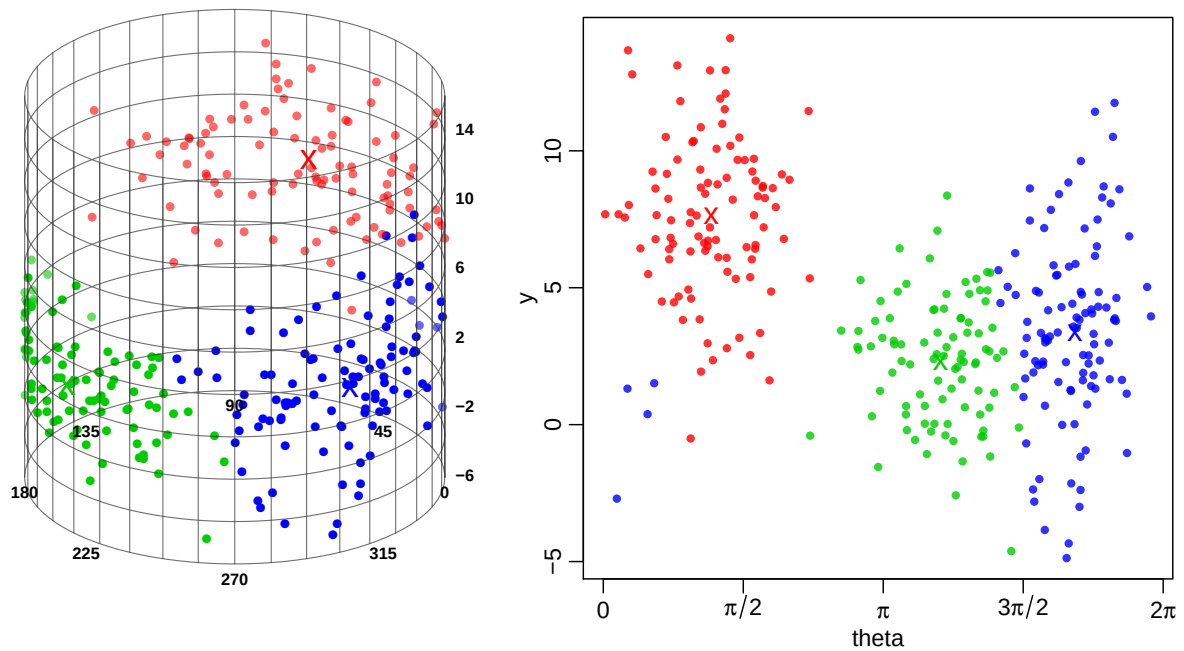
	Grupė 1	Grupė 2	Grupė 3					
N	102	102	96	N	100	5.3	5	3
π	0.364	0.341	0.295	μ_1	100	3.9	3	2.5
μ_θ	3.939	1.168	5.316	κ	100	1.2	4	7.5
μ_y	2.596	7.325	3.338	μ_2				
				σ				

Šis algoritmas taip pat suskirsto stebinius į grupes, kurių dydžiai yra arti 100. Priklausymo klasėms tikimybės yra apie 0,3, kas atitinka tikrovę. Palyginus gautus parametrų įverčius su pasirinktais modeliavimui, matome, kad skritulinio kintamojo vidutinės kryptis apvalinus sutampa, kaip ir praeito metodo. Tiesinio kintamojo grupių vidurkių įverčiai truputi skiriasi nuo pradinių, tačiau skirtumai yra nedideli.

Taigi abu EM klasterizavimo cilindriniai analogai vienodai gerai sugeba klasterizuoti duomenis, tačiau EMvMN algoritmas yra daug pranašesnis greičio prasme nei EM-SWG. Šitoje vietoje verta paminėti, kad nors EM-SWG ir lėtesnis, ji galima taikyti ir hipercilindriniais duomenimis, kas yra neįmanoma su EMvMN metodu.

2.4.2. Klasterizavimas naudojant k-vidurkių metodą

Dabar pritaikome kito klasterizavimo – k-vidurkių algoritmo cilindrinis analogus. Iš pradžių pritaikome fiksuotą cyk-vidurkių klasterizavimo metodą sugeneruotiems cilindriniais duomenimis. Šio metodo grupavimo rezultatus pavaizduojame grafiškai ant cilindro bei plokštumoje.



34 pav. Cilindrinų duomenų klasterizavimas naudojant fiksuotą cik-vidurkių metodą

Šis metodas iki konvergavimo užtruko 5 iteracijas, kas yra daug greičiau nei abu cilindriniai EM algoritmai. Jis taip pat, kaip ir EM metodai, sugebėjo gerai išskirti 3 grupes. Beje, šie du metodai duoda panašų stebinių grupavimą.

Įvertiname gautų grupių cilindrinis vidutinius vektorius ir grupių dydžius.

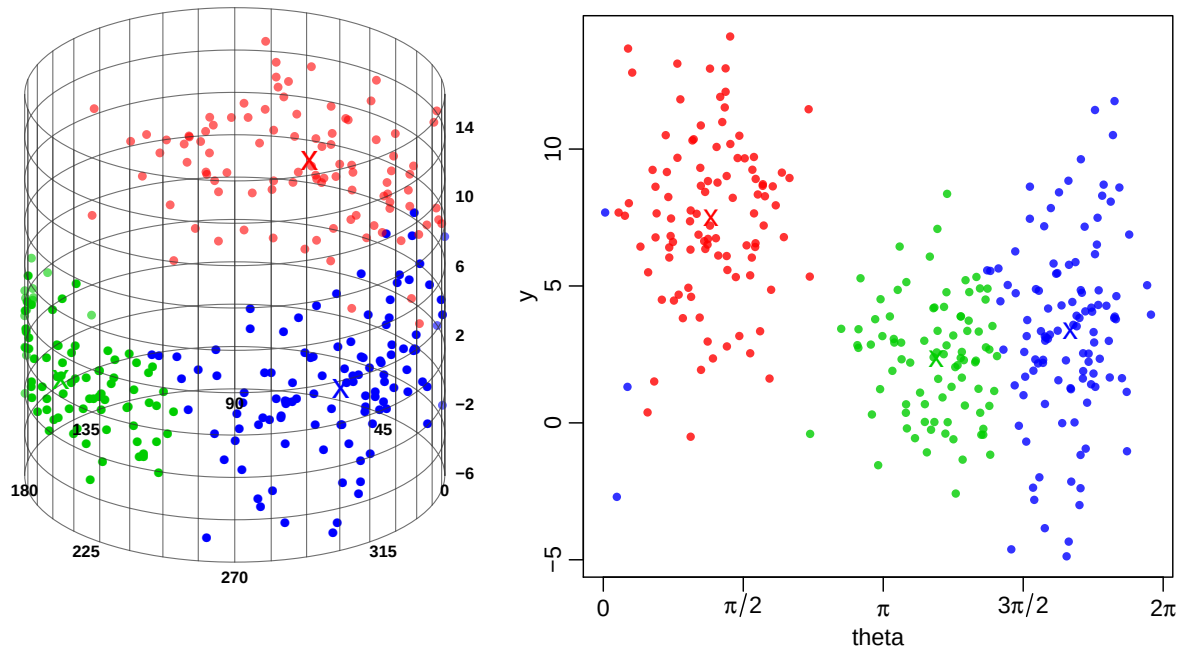
22 lentelė. Fiksuoto cik-vidurkių metodo klasterizavimo rezultatai: klasterių dydžiai ir centrai (kairėje), tikrų grupių modeliavimo parametrai iš lentelės 2 (dešinėje)

N	$\bar{\theta}$	$\bar{y}$	N	μ_1	κ	μ_2	σ
98	1.211	7.632	100	5.3	5	3	4
93	3.787	2.311	100	3.9	3	2.5	2
109	5.294	3.352	100	1.2	4	7.5	3

Matome, kad klasterių dydžiai yra arti 100, kas atitinka sumodeliuotų duomenų grupių dydžius, tačiau grupių dydžiai labiau skiriasi nuo tikrovės, nei EM algoritmų. Palyginus gautus parametrų įverčius su pasirinktais modeliavimui, matome, kad skritulinio kintamojo vidutinės kryptis pirmai ir trečiai grupei sutampa, o antros grupės skiriasi nedaug. Tiesinio kintamojo vidurkių įverčiai taip pat skiriasi nuo parinktų, tačiau nežymiai. Vadinasi fiksuotas cik-vidurkių klasterizavimo būdas yra geras metodas kryptinių duomenų grupavimui.

Toliau pritaikome cik-vidurkių klasterizavimo algoritmą be fiksavimo tiems patiems duomenims. Šio metodo klasterizavimo rezultatas yra pavaizduotas žemiau ant cilindro

bei plokštumoje.



35 pav. Cilindrinį duomenų klasterizavimas naudojant cik-vidurkių metodą

Šis algoritmas, kad pasiektų konvergavimo, užtruko šiek tiek daugiau iteracijų – 8, nei fiksuotas cik-vidurkių metodas. Tačiau šis algoritmas vis tiek yra žymiai greitesnis nei aukščiau taikyti cilindriniai EM klasterizavimo analogai. Iš grafiko matome, kad šis metodas sugeba gerai išskirti 3 grupes. Šio metodo grupės yra panašios į fiksuoto cik-vidurkių metodo gautas grupes, bet jos truputi skiriasi.

Įvertiname gautų grupių cilindrinis vidutinius vektorius ir grupių dydžius.

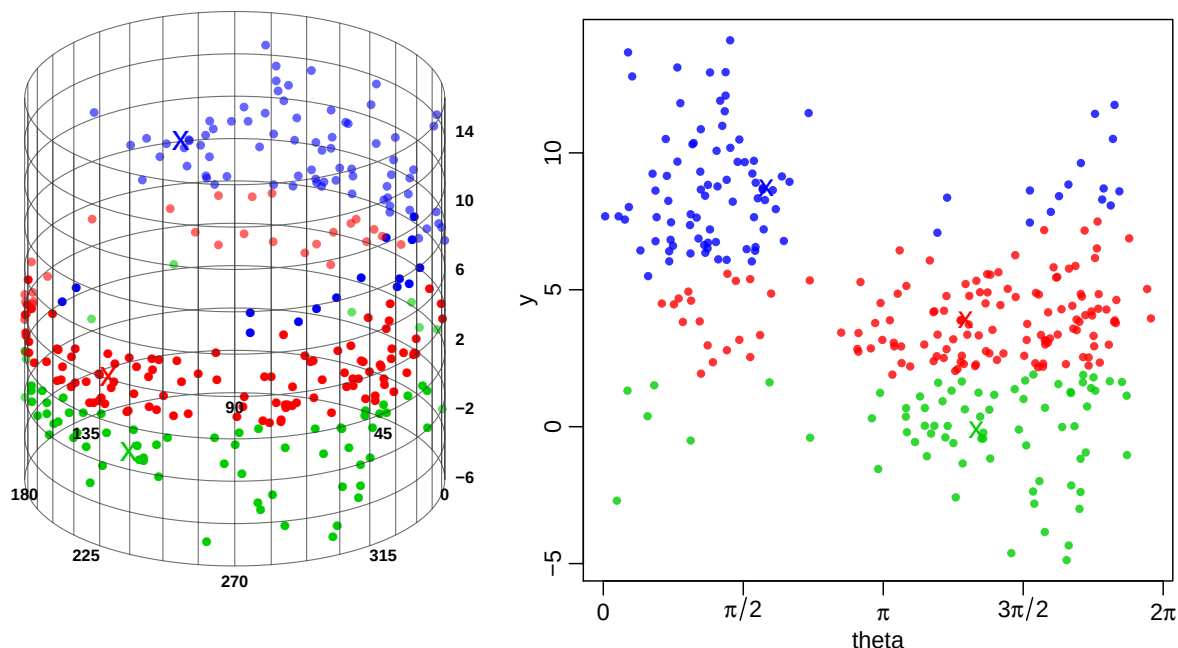
23 lentelė. cik-vidurkių metodo klasterizavimo rezultatai: klasterių dydžiai ir centrai (kairėje), tikrų grupių modeliavimo parametrai iš lentelės 2 (dešinėje)

N	$\bar{\theta}$	$\bar{y}$	N	μ_1	κ	μ_2	σ
99	1.207	7.496	100	5.3	5	3	4
87	3.737	2.350	100	3.9	3	2.5	2
114	5.242	3.348	100	1.2	4	7.5	3

Iš lentelės matome, kad klasterių dydžiai yra arti 100, tačiau grupių dydžiai labiau skiriasi nuo tikrovės, nei visų kitų anksčiau taikytų algoritmų. Lyginant gautus parametrų įverčius su pasirinktais modeliavimui bei kitų metodų gautais, matome, kad rezultatas yra analogiškas kaip ir fiksuoto cik-vidurkių metodo – skritulinio kintamojo vidutinės kryptis beveik sutampa su tikrosiomis reikšmėmis, o tiesinio kintamojo vidurkiai skiriasi, bet nežymiai, nuo pradinių.

2.4.3. Palyginimas su klasikiniu klasterizavimu

Toliau pritaikysime klasikinį k-vidurkių klasterizavimo algoritmą cilindriniams duomenims, kad įsitikintume, kad toks modeliavimo būdas yra klaidingas. Klasterių skaičius, kaip ir anksčiau, yra pasirenkamas lygus 3. Gautus rezultatus vaizduojame grafiškai.



36 pav. Cilindrinų duomenų klasterizavimas naudojant klasikinį k-vidurkių metodą

Toks algoritmas užtruko tik 2 iteracijas, kad pasiektų konvergavimą. Tačiau, iš grafiko matome, kad toks algoritmas nesugeba suskirstyti duomenis į teisingas grupes. Klasikinis k-vidurkių metodas suskirstė duomenis į grupes tik tai pagal tiesinį kintamąjį ir ignoravo skritulinį kintamąjį.

Žemiau yra pateikiami šio klasterizavimo grupių dydžiai bei centrai.

24 lentelė. Klasikinio k-vidurkių metodo klasterizavimo rezultatai: klasterių dydžiai ir centrai (kairėje), tikrų grupių modeliavimo parametrai iš lentelės 2 (dešinėje)

N	$\bar{\theta}$	$\bar{y}$	N	μ_1	κ	μ_2	σ
136	4.068	3.913	100	5.3	5	3	4
72	4.184	-0.111	100	3.9	3	2.5	2
92	1.831	8.739	100	1.2	4	7.5	3

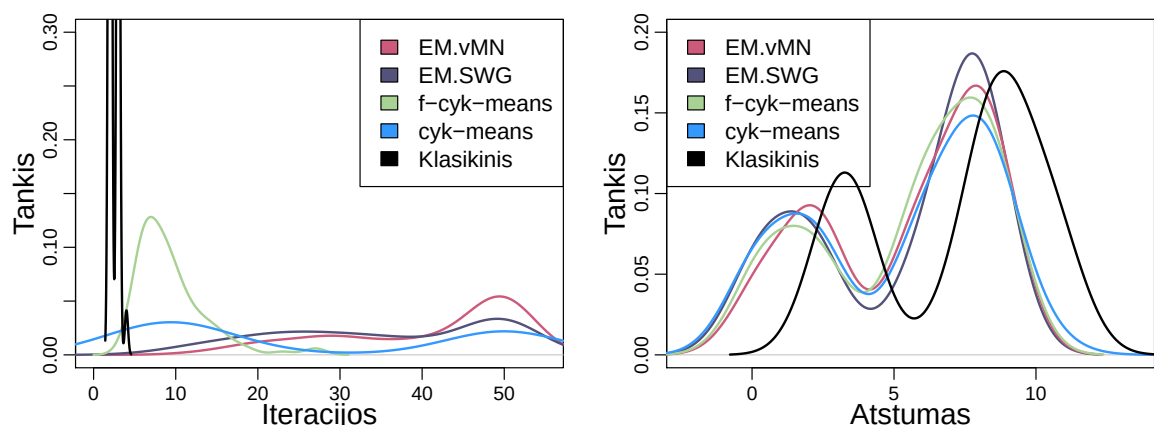
Matome, kad grupių dydžiai neatitinka tikrovės. Palyginus gautus parametrų įverčius su pasirinktais modeliavimui, matome, kad grupių centrai taip pat neatitinka tikrovės.

Taigi gavome analogiška rezultatą kaip ir sferinių duomenų atveju – klasikinių klasterizavimo metodų taikymas kryptiniams duomenims yra klaidingas. Šis pavyzdys parodo, kodėl prieš taikant klasikinius algoritmus kryptiniams duomenims, juos būtina modifikuoti, kad jie atsižvelgtų į tokių duomenų ypatumą.

2.4.4. Metodų tarpusavyje palyginimas

Šitame skyrelyje skirtingi klasterizavimo ant cilindro metodai yra palyginami tarpusavyje. Tam yra naudojamos duomenų pakartotinis modeliavimas ir jų klasterizavimas. Cilindriniai duomenys yra modeliuojami, taip pat kaip buvo aprašyta skyriuje 2.1.1. Modeliavimas yra kartojamas, skirtingai nuo sferinio atvejo, tik 100 kartų, nes kai kurių cilindrinio klasterizavimo algoritmų veikimo greitis yra pakankamai lėtas. Kiekvienu atveju yra pritaikome penki klasterizavimo metodai: von Mises-Normalusis EM, SWG-EM, fiksuotas ciklų vidurkių, ciklų vidurkių ir klasikinis k-vidurkių.

Padarius tokias simuliacijas, yra įvertinami gautų klasterių centrų cilindriniai atstumai iki originalių grupių vidutinių kryptų. Jie yra pateikti tankių grafikuose apačioje. Taip pat yra pateikiami kiekvieno pritaikyto metodo iteracijų iki konvergavimo skaičius.



37 pav. Klasterizavimo ant cilindro metodų charakteristikų tankiai: iteracijų skaičius ir cilindriniai atstumai tarp klasterių centrų ir originalių parametrų

Iš tankių grafikų matome, kad greičiausias klasterizavimo algoritmas yra klasikinis k-vidurkių metodas. Paskui pagal greitį eina ciklų vidurkių algoritmai – fiksuotas ciklų vidurkių algoritmas yra greitesnis. Be to, galima pastebėti, kas paprastas ciklų vidurkių algoritmas ne visada konverguoja per 50 iteracijų bei verta paminėti, kad iš 100 atvejų 1 kartą duomenys nebuvo suklasterizuoti į 3 grupes. Cilindriniai EM klasterizavimo algoritmai yra lėčiau iš visų nagrinėjamų – von Mises-Normalusis EM yra truputi lėtesnis nei SWG-EM algoritmas ir jis ne visada pasiekia konvergavimą per 50 iteracijų.

Lyginant cilindrinio klasterizavimo algoritmų tikslumą, matome, kad visi metodai, išskyrus klasikinį k-vidurkių metodą, turi panašius atstumus nuo suklasterizuotų grupių

centrų iki originalių grupių vidutinės krypties, o klasikinis k-vidurkių metodas turi didesnius nukrypimus.

Įvertiname aukščiau pateiktų charakteristikų vidurkius kiekvienam metodui.

25 lentelė. Klasterizavimo ant cilindro metodų vidutinės charakteristikos: iteracijų skaičius ir cilindriniai atstumas tarp klasterių centrų ir originalių parametrų

	vMN-EM	SWG-EM	Fix cyk-vidurkių	cyk-vidurkių	Klasikinis
iteracijos	40.84	35.63	9.42	25.69	2.45
atstumas	5.461	5.343	5.459	5.298	7.247

Iš pateiktos lentelės galima padaryti panašias išvadas kaip ir iš aukščiau pateiktų tankių grafikų. Nors greičiausias algoritmas yra klasikinis k-vidurkių klasterizavimas, jo tikslumas yra mažas, todėl šis algoritmas nėra geras metodas cilindrinį duomenų klasterizavimui. Abu cilindriniai EM klasterizavimo analogai yra daug lėtesni už kitus metodus, o jų tikslumas skiriasi nedaug, todėl naudoti juos cilindrinį duomenų klasterizavimui taip pat neverta. Palyginus du k-vidurkių metodo cilindrinis analogus, matome, kad fiksuotas cyk-vidurkių algoritmas yra dvigubai greitesnis, o jų klasterizavimo tikslumas skiriasi nedaug, todėl geriausias metodas cilindrinį duomenų klasterizavimui yra fiksuotas cyk-vidurkių algoritmas.

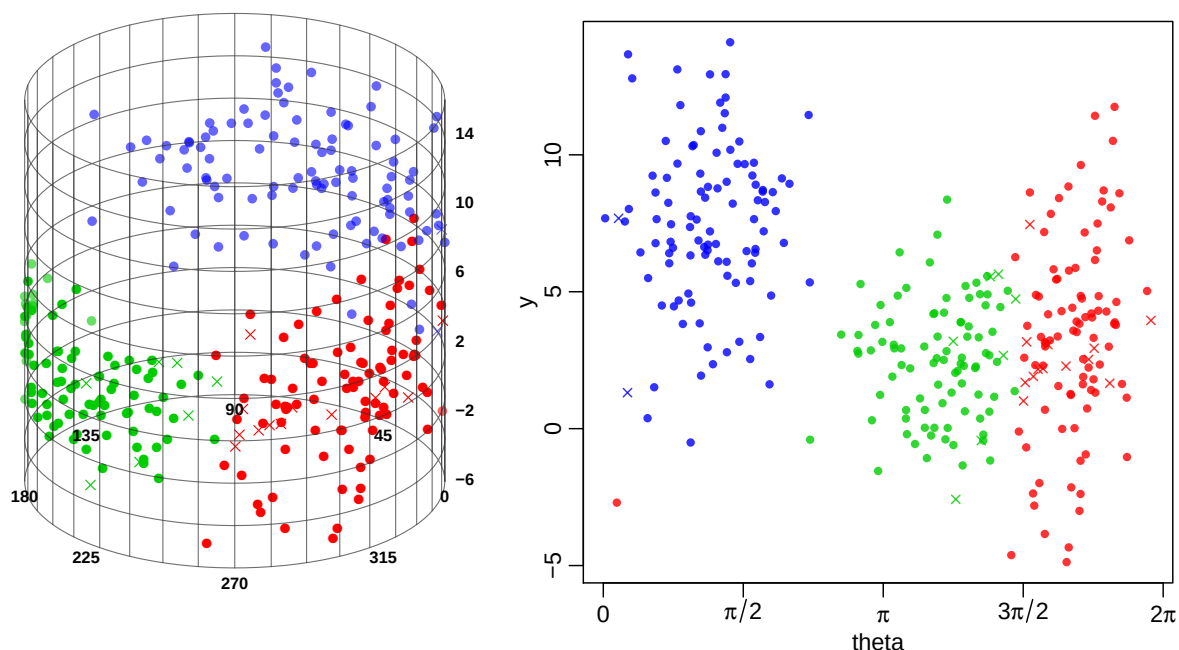
2.5. Cilindrinį duomenų klasifikavimas

Šiame skyrelyje, analogiškai kaip ir sferinių duomenų atveju, yra analizuojami sumodeliuoti bei realūs (meteorologiniai) cilindriniai duomenys. Šiems duomenims yra taikomi keli cilindrinio klasifikavimo metodai: von Mises-Normalioji diskriminantinė analizė, SWG kvadratinė diskriminantinė analizė bei logistinė regresija. Visų šių algoritmų veikimas yra lyginamas tarpusavyje bei su klasikine logistine regresija.

2.5.1. Diskriminantinė analizė

Šitame skyrelyje cilindrinės diskriminantinės analizės metodo analogai: von Mises-Normalioji diskriminantinė analizė ir SWG diskriminantinė analizė, yra pritaikomi sumodeliuotiems cilindriniam duomenims bei realiems originaliems bei dieniniams meteorologiniams duomenims.

Von Mises-Normalioji diskriminantinė analizė. Iš pradžių pritaikome von Mises-Normaliosios diskriminantinės analizės metodą sumodeliuotiems cilindriniam duomenims, kurie turi tris klases. Rezultatas yra pavaizduotas žemiau ant cilindro bei plokštumoje. Taškiukai žymi stebinius, kurie buvo gerai suklasifikuoti, o kryžiuukai – stebinius, kurie buvo priskirti netinkamoms klasėms.



38 pav. Klasifikavimas naudojant cilindrinę diskriminantinę von Mises-Normaliąją analizę ant sugeneruotų duomenų

Iš grafiko matome, kad dauguma stebinių buvo teisingai suklasifikuoti.

Kad patikrintume kaip gerai veikia toks klasifikavimo algoritmas, įvertiname klasifikavimo tikslumo matricą ir apskaičiuojame kitas klasifikavimo gerumo metrikas: tikslumą, specifiškumą, jautrumą ir AUC. Šios klasifikavimo gerumo metrikos yra skaičiuojamos kiekvienai klasei atskirai ir visoms kartu.

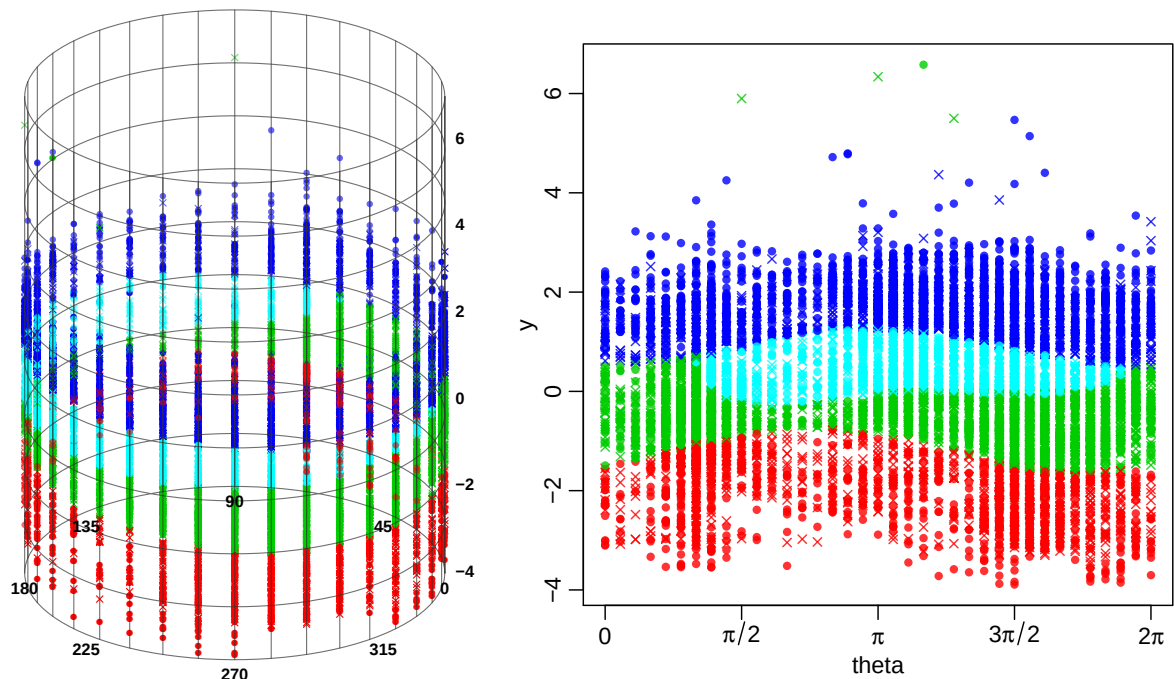
26 lentelė. Diskriminantinės von Mises-Normaliosios analizės ant sugeneruotų duomenų klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC	Prognozēs				
2	0.927	0.875	0.910	0.922	Grupēs	2	3	4	
3	0.937	0.926	0.880	0.922		2	91	7	2
4	0.990	0.980	0.990	0.990		3	12	88	0
viso	0.927	0.927	0.927	0.945		4	1	0	99

Šio metodo tikslumas šiems sumodeliuotiems duomenims yra 0,927, kas reiškia, kad toks algoritmas gerai gali prognozuoti grupes. Klasifikavimo tikslumo matrica taip pat parodo, kad dauguma stebinių buvo suklasifikuoti teisingai. Labiausiai painiojasi grupės su numeriais 2 ir 3 (raudona ir žalia), kurios yra išdėstytos šalia viena kitos, todėl tai natūralu.

Toliau pritaikome tą pati cilindrinę von Mises-Normaliosios diskriminantinės analizės metodą klasifikuojant meteorologinius duomenis į keturias grupes pagal metų laikus. Kadangi šis metodas tinka tik vienmačiams cilindriniais duomenimis, t. y. gali būti tik vienas tiesinis kintamasis, modeliavimas daromas naudojant pirmą pagrindinę komponentę, kaip iš grafikų braižyme.

Tokio algoritmo klasifikavimo rezultatas yra pateiktas žemiau ant cilindro bei plokštumoje. Kaip ir anksčiau taškiukai ir kryžiukai atitinkamai žymi gerai bei blogai suklasifikuotus stebinius.



39 pav. Meteorologinių duomenų klasifikavimas naudojant cilindrinę diskriminantinę von Mises-Normaliąją analizę

Palyginus šį grafiką su tikrosiomis grupėmis (17 pav.), matome, kad yra pakankamai daug duomenų, kurie yra netinkamai suklasifikuoti. Taip yra dėl to, kad von Mises-Normaliosios diskriminantinės analizės metode yra daroma prielaida, kad duomenys yra pasiskirstę pagal von Mises-Normalųjį pasiskirstymą, o iš grafikų akivaizdų, kad taip nėra.

Kaip ir sumodeliuotų duomenų atveju, įvertiname klasifikavimo tikslumo matricą ir kitus 4, aukščiau minėtus, klasifikavimo gerumo matus.

27 lentelė. Meteorologinių duomenų naudojant cilindrinę diskriminantinę von Mises-Normaliąją analizę klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC	Prognozès					
Win	0.788	0.556	0.457	0.673		Win	Spr	Sum	Aut	
Spr	0.667	0.442	0.512	0.622	Grupès	Win	958	845	61	230
Sum	0.864	0.662	0.869	0.865		Spr	592	1343	326	363
Aut	0.743	0.433	0.288	0.586		Sum	0	78	1845	201
viso	0.531	0.523	0.531	0.686		Aut	173	775	554	607

Klasifikavimo tikslumo matrica bei kitos metrikos parodo, kad tik tai apie pusę duomenų yra tinkamai suklasifikuoti. Bendras šios cilindrinės diskriminantinės analizės metodo tikslumas yra 0,531.

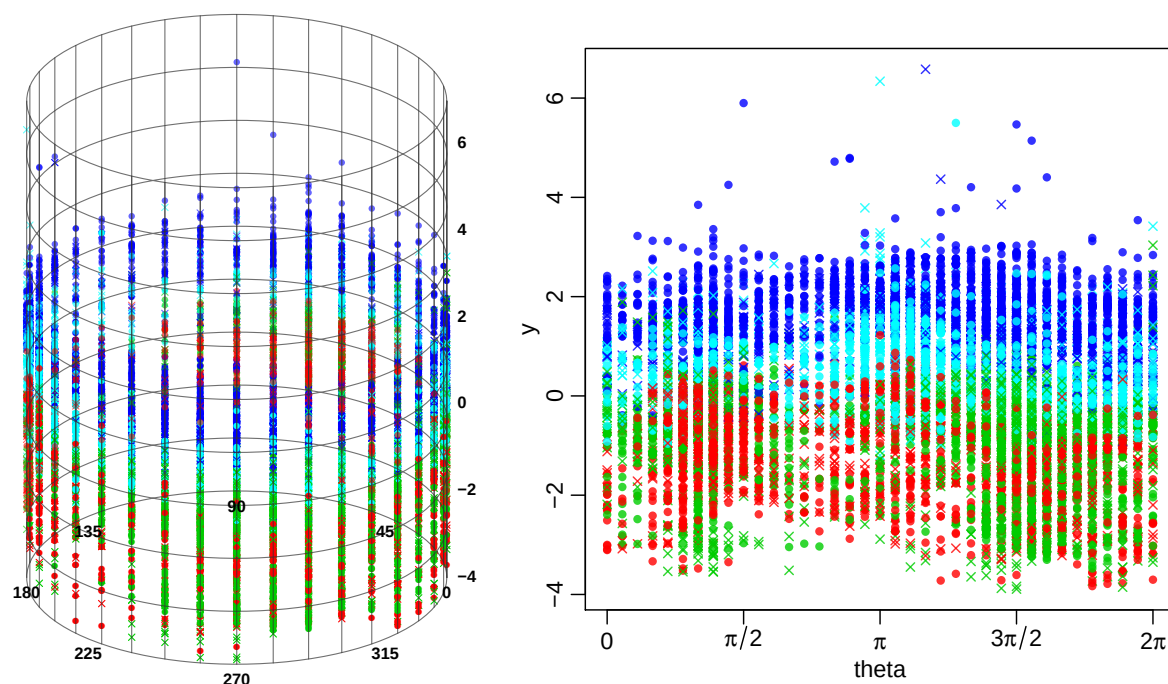
Norint pagerinti šio modelio tikslumą, pritaikome von Mises-Normaliąją diskriminantinę analizę ne pagrindiniai komponentai, o kiekvienam tiesiniam kintamajam atskirai. Po to, gautos prognozės yra transformuojamos į vieną naudojant klasikinį logistinės regresijos modelį. Šio modelio parametrus patikriname hipotezę apie jų reikšmingumą. Hipotezės p-reikšmės yra pateiktos lentelėje žemiau.

28 lentelė. hipercilindrinės diskriminantinės analizės ant meteorologinių duomenų parametrų reikšmingumo hipotezės minimalios p-reikšmės

	Spr	Sum	Aut
(Intercept)	1.493e-16	0.0348	< 1e-16
Temp	< 1e-16	< 1e-16	< 1e-16
Dew.temp	< 1e-16	< 1e-16	< 1e-16
Hum	< 1e-16	< 1e-16	3.864e-08
Precip	0.2976	0.0212	0.0170
Wind.spd	7.349e-08	< 1e-16	8.664e-06
Press	1.180e-09	0.0002	1.314e-14

Kaip matome, visi parametrai yra reikšmingi šiame modelyje, tačiau pačiu nereikšmingiausiu galima laikyti kintamąjį *Precip*.

Pavaizduojame tokio hipercilindrinų duomenų klasifikavimo metodo rezultatus grafiškai ant cilindro bei plokštumoje.



40 pav. Meteorologinių duomenų klasifikavimas naudojant hipercilindrinę diskriminantinę von Mises-Normaliąją analizę

Taip pat, kad patikrintume kaip gerai veikia toks klasifikavimo algoritmas, įvertiname klasifikavimo tikslumo matricą, tikslumą, specifiškumą, jautrumą bei AUC.

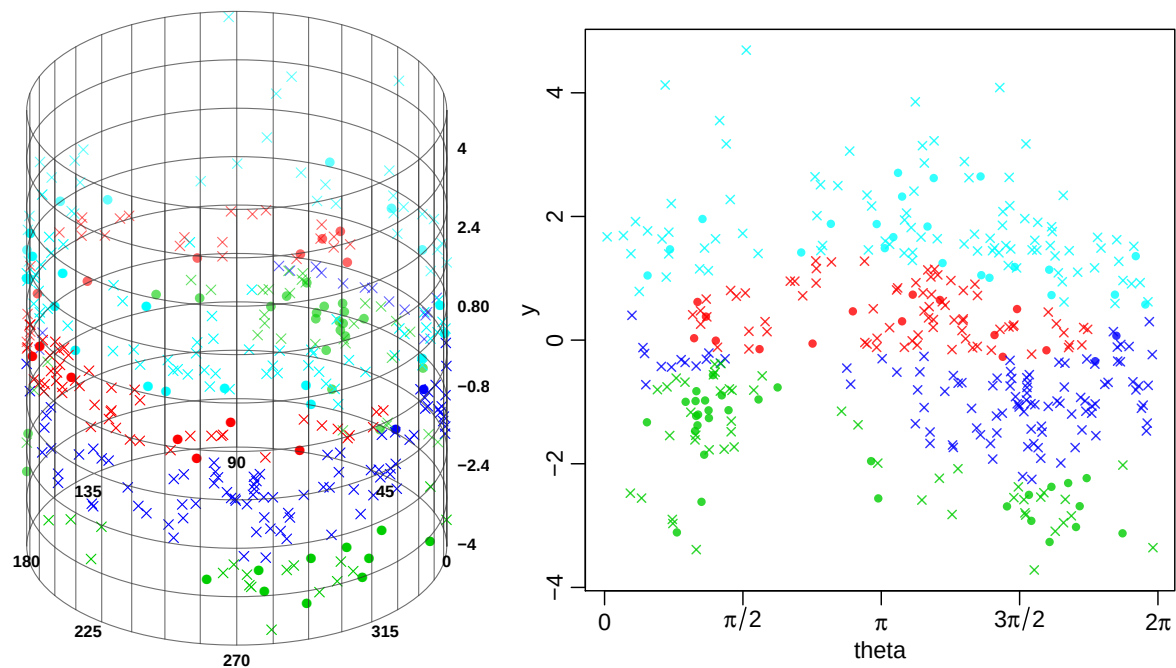
29 lentelė. Meteorologinių duomenų naudojant hipercilindrinę diskriminantinę von Mises-Normaliąją analizę klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC	Prognozès					
Win	0.860	0.715	0.670	0.794	Grupès	Win	Spr	Sum	Aut	
Spr	0.773	0.605	0.645	0.735		Win	1404	608	0	82
Sum	0.904	0.751	0.893	0.900		Spr	325	1693	206	400
Aut	0.794	0.579	0.457	0.677		Sum	0	10	1896	218
viso	0.666	0.663	0.666	0.777	Aut	235	486	424	964	

Klasifikavimo tikslumo matrica parodo, kad dauguma stebinių yra gerai suklasifikuota, tačiau vis tiek yra pakankamai daug atvejų, kai stebinis yra priskiriamas netinkamai klasei. Bendras šio metodo tikslumas yra 0,666, kas yra geriau nei von Mises-Normaliosios diskriminantinės analizės metodas su viena pagrindine komponente.

Toliau tą pačią analizę pritaikome dieniniams meteorologiniams duomenims. Pritaikome cilindrines von Mises-Normaliosios diskriminantinės analizės algoritmą naudojant pirmą pagrindinę komponentę klasifikuojant duomenis į keturias grupes pagal metų laikus. Tokio algoritmo klasifikavimo rezultatas yra pateiktas žemiau ant cilindro bei plokštumo-

je. Kaip ir anksčiau taškiukai ir kryžiuikai atitinkamai žymi gerai bei blogai suklasifikuotus stebinius.



41 pav. Dieninių meteorologinių duomenų klasifikavimas naudojant cilindrinę diskriminantinę von Mises-Normaliąją analizę

Iš grafiko matome, kad beveik visi stebiniai yra netinkamai suklasifikuoti, o pats vaizdas yra panašus į originaliu duomenų klasifikavimą naudojant tą patį metodą.

Įvertiname šio metodo klasifikavimo tikslumo matricą, tikslumą, specifiškumą, jautrumą bei AUC.

30 lentelė. Dieninių meteorologinių duomenų naudojant cilindrinę diskriminantinę von Mises-Normaliąją analizę klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC	Prognozės				
						Win	Spr	Sum	Aut
Spr	0.665	0.392	0.277	0.550	Grupės	Win	14	37	34
Sum	0.499	0.019	0.022	0.335		Spr	15	31	49
Aut	0.582	0.193	0.242	0.464		Sum	18	2	2
Win	0.613	0.161	0.156	0.454		Aut	40	9	20
viso	0.179	0.191	0.174	0.451					

Bendras tikslumas šiems duomenims yra 0,179, kas yra blogiau nei atsitiktinis spėjimas. Klasifikavimo tikslumo matrica bei kitos metrikos taip pat parodo, kad dauguma stebinių yra priskiriami prie netinkamų grupių. Šis pavyzdys parodo, kad taikyti von

Mises-Normaliąją diskriminantinę analizę, kai grupių skirstinys nėra panašus į von Mises-Normalųjį ir grupės neatsiskiria vizualiai, neturi prasmės.

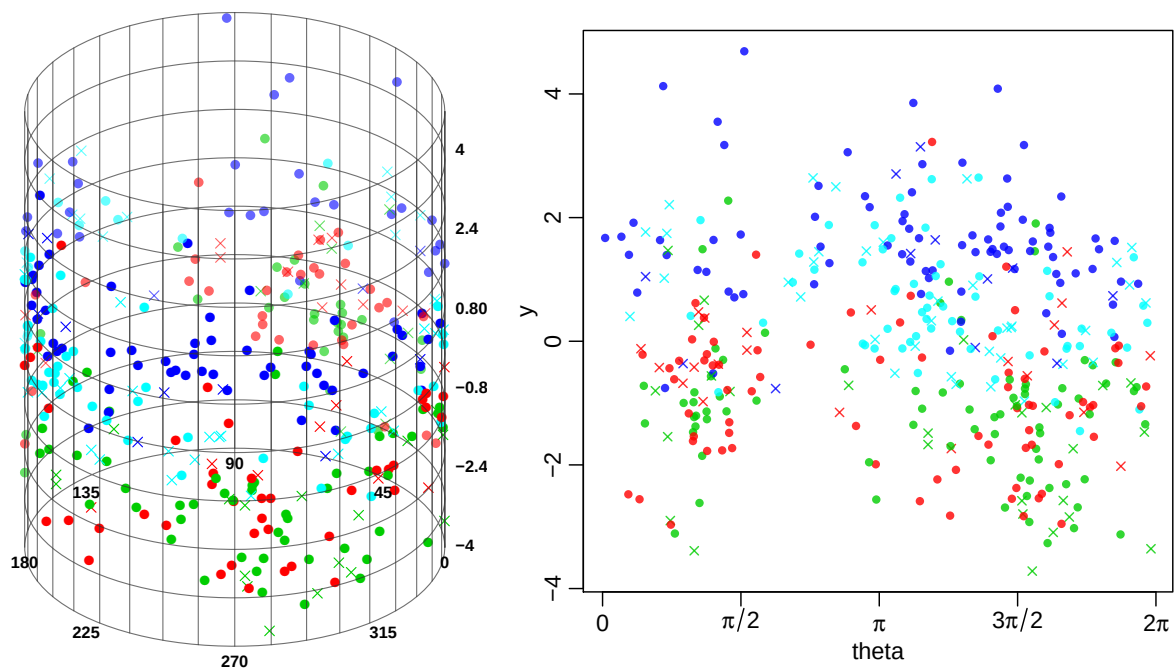
Kaip ir originaliu meteorologinių duomenų atveju, pritaikome von Mises-Normaliąją diskriminantinę analizę kiekvienam tiesiniam kintamajam atskirai ir sukombinuojam atskiras prognozes į vieną su logistinės regresijos modeliu. Tokio modelio parametrų reikšmingumas yra pateiktas žemiau.

31 lentelė. hipercilindrinės diskriminantinės analizės ant dieninių meteorologinių duomenų parametrų reikšmingumo hipotezės minimalios p-reikšmės

	Spr	Sum	Aut
(Intercept)	< 1e-16	< 1e-16	< 1e-16
Temp	< 1e-16	< 1e-16	< 1e-16
Dew.temp	< 1e-16	< 1e-16	< 1e-16
Hum	0.0064	0.0014	0.0149
Precip	0.1433	0.5253	0.1269
Wind.spd	0.0401	0.0003	0.0317
Press	0.1242	0.2395	0.0633

Dieniniams meteorologiniams duomenims toks modelis turi du nereikšmingus parametrus: *Precip* ir *Press*, visi likusieji 4 parametrai yra reikšmingi.

Pavaizduojame tokio hipercilindrinės diskriminantinės analizės modelio rezultatus grafiškai ant cilindro bei plokštumoje.



42 pav. Dieninių meteorologinių duomenų klasifikavimas naudojant hipercilindrinę diskriminantinę von Mises-Normaliąją analizę

Palyginus gauta grafiką su tikrosiomis grupėmis (20 pav.), matome, kad vaizdas yra panašus.

Kad įsitikintume modelio gerume, įvertiname klasifikavimo tikslumo matricą, tikslumą, specifiškumą, jautrumą bei AUC.

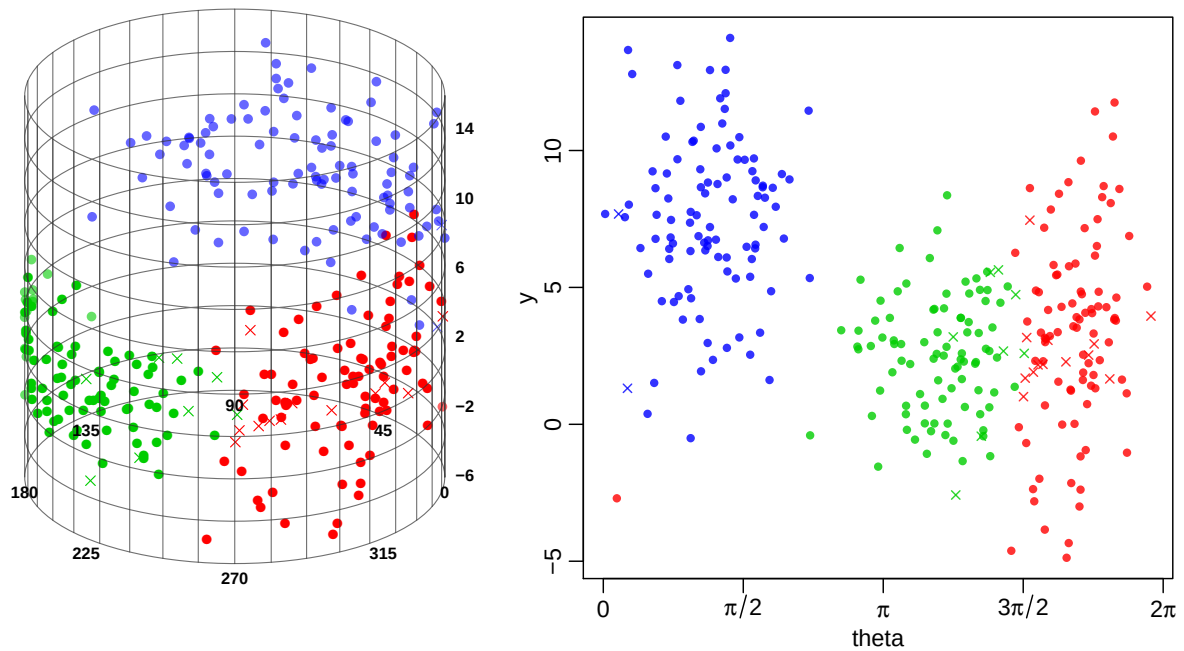
32 lentelė. Dieninių meteorologinių duomenų naudojant hipercilindrinę diskriminantinę von Mises-Normaliąją analizę klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC	Prognozès					
Spr	0.836	0.738	0.679	0.790	Grupès	Win	Spr	Sum	Aut	
Sum	0.945	0.874	0.902	0.931		Win	73	11	0	6
Aut	0.818	0.613	0.626	0.752		Spr	9	76	6	21
Win	0.901	0.777	0.811	0.870		Sum	0	0	83	9
viso	0.751	0.750	0.755	0.836	Aut	12	16	6	57	

Klasifikavimo tikslumo matrica parodo, kad dauguma stebinių yra priskiriami prie tinkamų grupių, o klaidingų priskyrimų palyginus yra nedaug. Bendros tokio modelio tikslumas dieniniams meteorologiniams duomenims yra 0,751, kas yra visai neblogai. Vadinasi taikyti von Mises-Normaliąją diskriminantinę analizę naudojant pirmą pagrindinę komponentę yra netinkama, nes dėl kombinavimo yra prarandama naudinga informacija apie duomenis.

SWG diskriminantinė analizė. Kadangi originalia von Mises-Normaliąją diskriminantinę analizę galima taikyti tik paprastiems cilindrinėms duomenims, o ne hipercilindrinėms, vėl nagrinėti kitą cilindrinę diskriminantinės analizės analogą – SWG diskriminantinę analizę.

Iš pradžių SWG diskriminantinės analizės metodas yra taikomas sumodeliuotiems cilindrinėms duomenims. Tokio klasifikavimo rezultatas yra pateiktas žemiau ant cilindro bei plokštumoje. Kaip ir anksčiau taškiukai žymi stebinius, kurie buvo gerai suklasifikuoti, o kryžiai – stebinius, kurie buvo priskirti netinkamoms klasėms.



43 pav. Klasifikavimas naudojant cilindrinę diskriminantinę SWG analizę ant sugeneruotų duomenų

Iš grafiko matome, kad dauguma stebinių buvo teisingai suklasifikuoti, kaip ir kitos diskriminantinės analizės atveju.

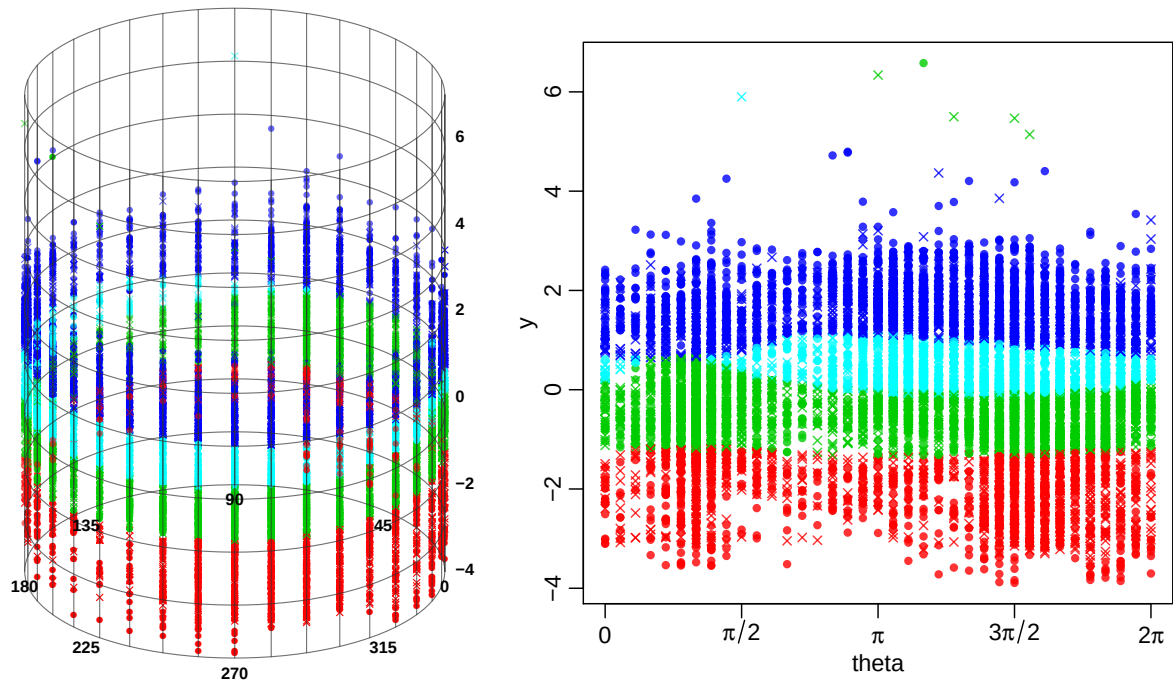
Kad patikrintume kaip gerai veikia toks klasifikavimo algoritmas, įvertiname klasifikavimo tikslumo matricą bei kitus klasifikavimo gerumo matavimus: tikslumą, specifiškumą, jautrumą bei AUC.

33 lentelė. Diskriminantinės SWG analizės ant sugeneruotų duomenų klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC	Prognozēs				
2	0.923	0.874	0.900	0.918	Grupēs	2	3	4	
3	0.933	0.917	0.880	0.920		2	90	8	2
4	0.990	0.980	0.990	0.990		3	12	88	0
viso	0.923	0.924	0.923	0.942		4	1	0	99

Klasifikavimo tikslumo matrica parodo, kad beveik visi stebiniai yra priskirti prie tinkamų grupių. Tarpusavyje painiojasi tik dvi grupės esančios šalia. Bendros SWG diskriminantinės analizės tikslumas yra 0,923, kas yra gerai. Palyginus šį rezultatą su von Mises-Normaliosios diskriminantinės analizės rezultatu, matome, kad šis metodas yra truputi blogesnis, bet nežymiai. Kadangi abiejų metodų rezultatų tikslumas yra apytiksliai lygus, verta palyginti algoritmus greičio prasme. Kaip buvo minėta teorinėje dalyje, SWG pasiskirstymo parametrai yra vertinami iteraciniu būdu, dėl to, SWG diskriminantinės analizės algoritmo greitis yra lėtesnis, nei von Mises-Normaliosios.

Pritaikome SWG diskriminantinės analizės metodą realiesiems meteorologiniams duomenims. Tokio metodo klasifikavimas yra pateikiamas žemiau ant cilindro bei plokštumoje.



44 pav. Meteorologinių duomenų klasifikavimas naudojant cilindrinę diskriminantinę SWG

Matome, kad toks rezultatas yra panašus į von Mises-Normaliosios diskriminantinės

analizės rezultata meteorologiniams duomenims, t. y. klasifikavimas yra netinkamas tokiems duomenims.

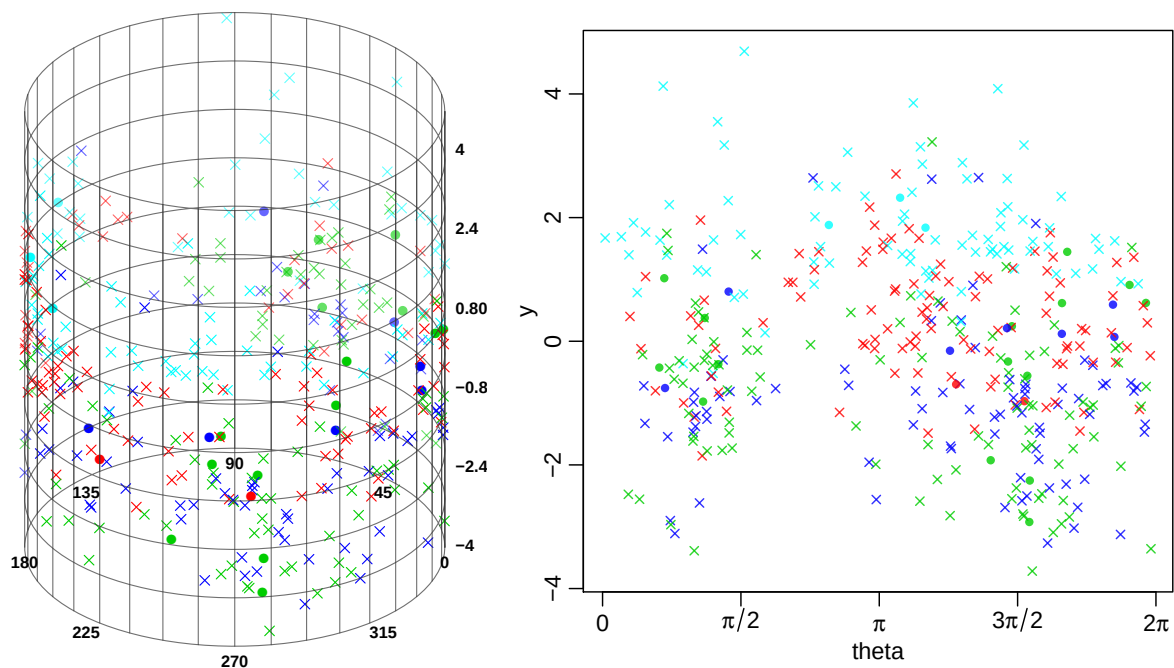
Kaip ir anksčiau įvertiname tokio metodo klasifikavimo tikslumo matricą, tikslumą, specifiškumą, jautrumą bei AUC.

34 lentelė. Meteorologinių duomenų naudojant cilindrinę diskriminantinę SWG klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC	Prognozės					
Win	0.793	0.568	0.485	0.686	Grupės	Win	Spr	Sum	Aut	
Spr	0.673	0.450	0.521	0.629		Win	1015	817	66	196
Sum	0.866	0.663	0.882	0.871		Spr	594	1366	322	342
Aut	0.752	0.458	0.284	0.590		Sum	0	81	1873	170
viso	0.542	0.535	0.543	0.694	Aut	179	769	562	599	

Klasifikavimo tikslumo matrica bei kiti gerumo matai parodo, kad tik tai apie pusę duomenų yra tinkamai suklasifikuoti. Bendras šios cilindrinės diskriminantinės analizės metodo tikslumas yra 0,542. Palyginus bendrą tikslumą su praeito metodo, šis metodas veikia truputi geriau. Tačiau, grupių parametrų vertinimas yra lėtas. Kad pasiektų konvergavimo algoritmui užtruko 18, 18, 12 ir 12 iteracijų atitinkamoms grupėms.

Pritaikome SWG diskriminantinės analizės metodą dieniniams meteorologiniams duomenims. Tokio metodo klasifikavimas yra pateikiamas žemiau ant cilindro bei plokštumoje.



45 pav. Dieninių meteorologinių duomenų klasifikavimas naudojant cilindrinę diskriminantinę SWG

Šiems duomenims grupių parametrų vertinimas atitinkami užtruks 1, 1, 14 ir 4 iteracijas iki konvergavimo. Grafikas parodo, kad beveik visi stebiniai yra netinkamai suklasifikuoti, kas ir vėl atitinka von Mises-Normaliosios diskriminantinės analizės rezultatus originaliems meteorologiniams duomenims.

Įvertiname tokio metodo klasifikavimo tikslumo matricą, tikslumą, specifiškumą, jautrumą bei AUC.

35 lentelė. Dieninių meteorologinių duomenų naudojant cilindrinę diskriminantinę SWG klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC	Prognozès					
Spr	0.496	0.134	0.134	0.389	Grupès	Win	Spr	Sum	Aut	
Sum	0.574	0.081	0.076	0.403		Win	2	86	2	0
Aut	0.561	0.036	0.033	0.379		Spr	22	15	68	7
Win	0.509	0.019	0.022	0.340		Sum	11	0	7	74
viso	0.070	0.068	0.066	0.378		Aut	68	11	9	3

Bendras tikslumas gavo 0,0701. Klasifikavimo tikslumo matrica bei kiti matai parodo, kad beveik visi stebiniai yra priskirti prie klaidingų grupių, tačiau algoritmas beveik visą laiką painioja tas pačias grupes, todėl ateityje tokį modelio prognozes galima koreguoti, kad padidinti klasifikavimo tikslumą. Jei pakeistume prognozių grupes vietomis,

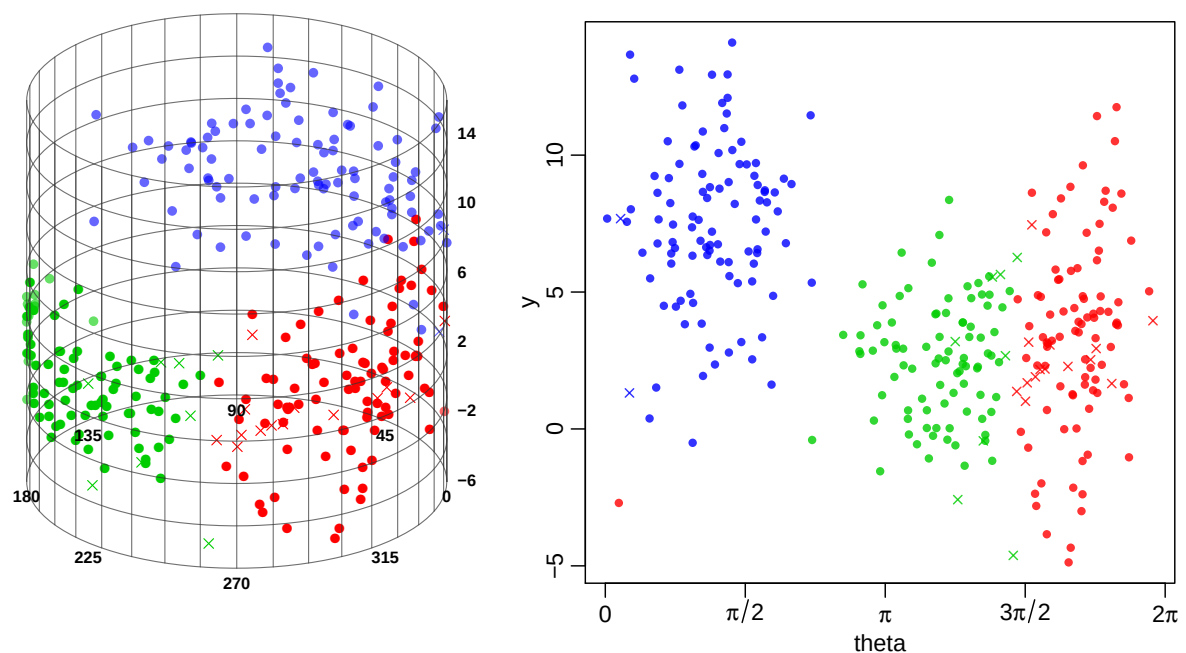
tai gautume tikslumą 0,769, bet naudoti tokią metodiką yra pavojinga, nes toks efektas galėjo gautis atsitiktinai.

Taigi iš cilindrinės diskriminantinės analizės metodų analogų, galima padaryti išvada, kad abu metodai veikia panašiai gerai/blogai, bet von Mises-Normaliosios diskriminantinės analizės metodas yra pranašesnis greičio prasme.

2.5.2. Logistinė regresija

Šitame skyrelyje cilindrinis multinominės logistinės regresijos modelis, kaip ir diskriminantinė analizė, yra taikomas norint suklasifikuoti sumodeliuotus cilindrinis bei meteorologinius duomenims.

Iš pradžių cilindrinės regresijos modelį pritaikome sumodeliuotiems duomenims, kurie turi 3 klases. Tokio klasifikavimo rezultatas yra pateiktas žemiau ant cilindro bei plokštumoje.



46 pav. Klasifikavimas naudojant cilindrinę logistinę regresiją ant sugeneruotų duomenų

Iš grafiko, matome, kad beveik visi taškai yra tinkamai suklasifikuoti. Kad įsitikintume šiuo teiginiu, įvertiname šio modelio gerumą skaitmeniškai. Tam vertiname klasifikavimo tikslumo matricą, tikslumą, specifiškumą, jautrumą bei AUC.

36 lentelė. Cilindrinė logistinė regresija ant sugeneruotų duomenų klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC
2	0.92	0.865	0.90	0.915
3	0.93	0.916	0.87	0.915
4	0.99	0.980	0.99	0.990
viso	0.92	0.920	0.92	0.940

Prognozès				
	2	3	4	
Grupès	2	90	8	2
	3	13	87	0
	4	1	0	99

Cilindrinės logistinės regresijos tikslumas sumodeliuotiems duomenims yra 0,92, kas parodo, kad modelis sugeba gerai prognozuoti grupes. Iš klasifikavimo tikslumo matricos, galime daryti tokia pačią išvadą. Dauguma stebinių yra priskiriama prie tinkamos klasės, o klaidos atsiranda tarp klasių, kurios yra išdėstytos šalia, kas yra natūralu. Palyginus gautą rezultatą su cilindrinės diskriminantinės analizės metodais, rezultatas yra gaunamas beveik toks pats.

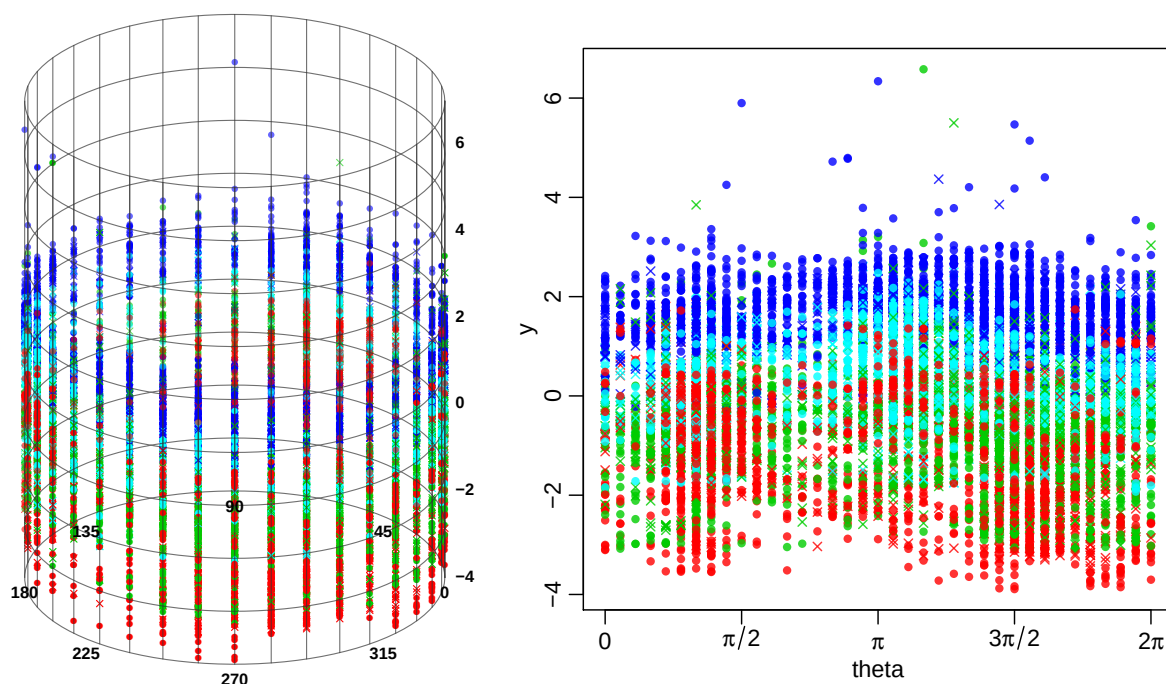
Toliau pritaikome cilindrinės logistinės regresijos modelį, norint klasifikuoti meteorologinius duomenis pagal metu laikus. Šiuo atveju analizė yra atliekama ant hipercilindrinų duomenų, todėl įvertiname kiekvieno parametro modelyje reikšmingumą. Hipotezės apie parametrų reikšmingumą p-reikšmės yra pateikiamos žemiau.

37 lentelė. Cilindrinės logistinės regresijos ant meteorologinių duomenų parametrų reikšmingumo hipotezės p-reikšmės

	Spr	Sum	Aut
(Intercept)	< 1e-16	< 1e-16	< 1e-16
cos(Wind.dir)	6.010e-10	< 1e-16	0.0003
sin(Wind.dir)	0.2304	1.511e-11	9.513e-07
Temp	< 1e-16	< 1e-16	< 1e-16
Hum	< 1e-16	0.1281	0.478
Precip	1.207e-06	0.0043	0.0285
Wind.spd	0.5981	< 1e-16	0.0024
Press	< 1e-16	< 1e-16	< 1e-16

Matome, kad šiame modelyje visi parametrai yra reikšmingi. Tačiau, kaip pačius nereikšmingiausius galima pažymėti: *Hum* ir *Wind.spd*. Palyginus su cilindrinės diskriminantinės analizės metodu, parametrų reikšmingumas skiriasi.

Pavaizduojame cilindrinės logistinės regresijos modelio klasifikavimo rezultatus grafiškai ant cilindro bei plokštumoje.



47 pav. Meteorologinių duomenų klasifikavimas naudojant cilindrinę logistinę regresiją

Palyginus duota grafiką su tikrosiomis grupėmis (17 pav.), galima tarti, kad šis modelis pakankamai neblogai gali klasifikuoti turimus duomenis.

Įvertiname šio modelio klasifikavimo tikslumo matricą, tikslumą, specifiskumą, jautrumą bei AUC.

38 lentelė. Meteorologinių duomenų naudojant cilindrinę logistinę regresiją klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC
Win	0.892	0.738	0.833	0.871
Spr	0.803	0.670	0.643	0.756
Sum	0.919	0.804	0.871	0.903
Aut	0.815	0.628	0.527	0.715
viso	0.714	0.710	0.719	0.811

		Prognozės			
		Win	Spr	Sum	Aut
Grupės	Win	1744	264	0	86
	Spr	427	1687	170	340
	Sum	0	42	1851	231
	Aut	193	524	281	1111

Iš klasifikavimo matricos galime spręsti, kad dauguma stebinių yra priskiriama prie teisingų grupių, tačiau klaidingų priskyrimų irgi yra pakankamai daug. Bendras tokio modelio klasifikavimo tikslumas yra 0,714, kas yra geriau už bet kurio cilindrinės diskriminantinės analizės metodo gautą tikslumą. Taip yra dėl to, kad šis modelis nedaro jokių išankstinių prielaidų apie duomenų pasiskirstymą.

Galiausiai pritaikome cilindrinės logistinės regresijos modelį dieniniams meteorologiniams duomenims. Kaip ir anksčiau, patikriname hipotezę apie parametrų reikšmingumą

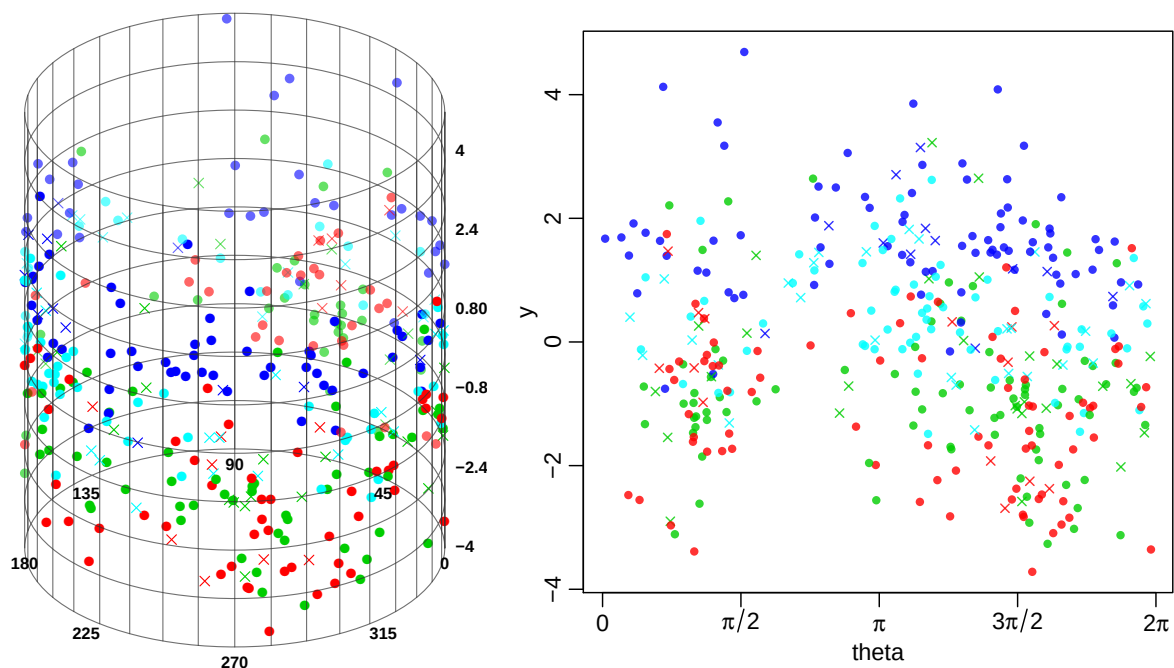
ir pateikime atitinkamas p-reikšmes.

39 lentelė. Cilindrinės logistinės regresijos ant dieninių meteorologinių duomenų parametrų reikšmingumo hipotezės p-reikšmės

	Spr	Sum	Aut
(Intercept)	< 1e-16	< 1e-16	< 1e-16
cos(Wind.dir)	0.3728	0.0012	0.5982
sin(Wind.dir)	0.6937	0.5784	0.5605
Temp	3.223e-10	< 1e-16	1.851e-14
Hum	6.885e-11	0.0004	0.0303
Precip	0.0072	0.2725	0.5373
Wind.spd	0.8409	0.0657	0.4013
Press	< 1e-16	< 1e-16	< 1e-16

Šiame modelyje yra vienas nereikšmingas parametras – *Wind.spd*, kas taip pat kaip ir praeito modelio, skiriasi nuo cilindrinės diskriminantinės analizės gauto rezultato. Taip pat verta paminėti, kad sekantys nereikšmingiausi parametrai modelyje yra *Wind.dir* ir *Precip*.

Pavaizduojame cilindrinės logistinės regresijos gautus klasifikavimo rezultatus grafiškai ant cilindro bei plokštumoje.



48 pav. Dieninių meteorologinių duomenų klasifikavimas naudojant cilindrinę logistinę regresiją

Palyginus duota grafiką su tikrosiomis grupėmis (20 pav.), galima tarti, kad šis modelis pakankamai neblogai gali klasifikuoti turimus duomenis.

Įvertiname šio modelio klasifikavimo tikslumo matricą, tikslumą, specifiškumą, jautrumą bei AUC.

40 lentelė. Dieninių meteorologinių duomenų naudojant cilindrinę logistinę regresiją klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC	Prognozės				
						Win	Spr	Sum	Aut
Spr	0.855	0.755	0.741	0.821	Grupės				
Sum	0.945	0.874	0.902	0.931		Win	79	6	0
Aut	0.849	0.690	0.659	0.784		Spr	9	83	6
Win	0.935	0.849	0.878	0.915		Sum	0	1	83
viso	0.792	0.792	0.795	0.863		Aut	5	20	6

Bendras tokio modelio tikslumas šiems duomenims yra 0,792. Čia geriausias gautas rezultatas klasifikuojant dieninius meteorologinius duomenis. Klasifikavimo matrica parodo, kad daugumoje atvejų priskiriama klasė yra teisinga. Daugiausiai klaidų yra daroma tarp pavasario ir rudens grupių.

Taigi atlikus šią analizę, galima daryti išvada, kad cilindrinės logistinės regresijos modelis yra geresnis klasifikavimo metodas nei cilindriniai diskriminantinės analizės analogai. Taip yra dėl to, kad toks modelis yra lankstesnis, nes nedaro jokių prielaidų apie duomenų pasiskirstymą. Be to, verta paminėti, kad šis modelis tinka ir hipercilindriniais duomenims, ir duomenims turintiems kelis skritulinius ir kelis tiesinius kintamuosius.

2.5.3. Palyginimas su klasikiniu klasifikavimu

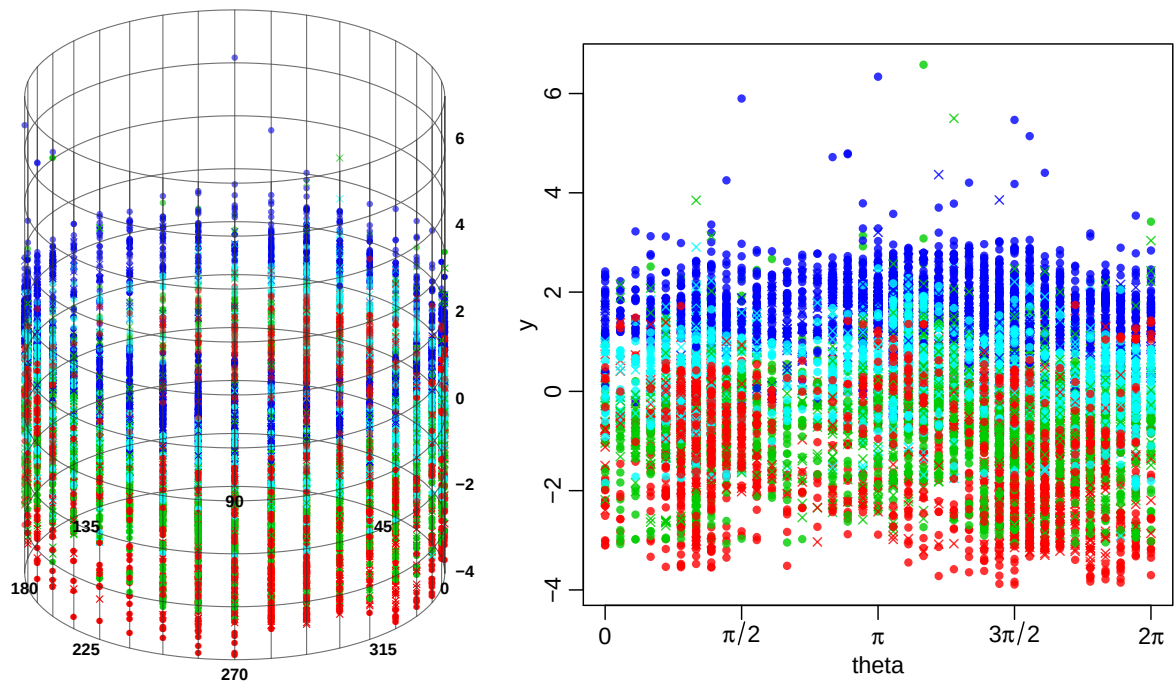
Šioje dalyje klasikinis regresijos modelis taikomas tik tai realiesiems meteorologiniams duomenims. Kaip ir cilindrinės logistinės regresijos atveju, iš pradžių pateikiame modelio parametrų reikšmingumo p-reikšmes.

41 lentelė. Klasikinės logistinės regresijos ant meteorologinių duomenų parametų reikšmingumo hipotezės p-reikšmės

	Spr	Sum	Aut
(Intercept)	< 1e-16	< 1e-16	< 1e-16
Temp	< 1e-16	< 1e-16	< 1e-16
Hum	< 1e-16	0.2527	0.3714
Precip	1.535e-07	0.002	0.0309
Wind.dir	0.1807	9.859e-10	0.0038
Wind.spd	0.6886	< 1e-16	0.0001
Press	< 1e-16	< 1e-16	< 1e-16

Matome, kad visi modelio parametrai yra reikšmingi, tačiau kaip pačius nereikšmingiausius galima išrinkti *Wind.dir* ir *Wind.spd*.

Pavaizduojame tokio klasifikavimo rezultatus grafiškai ant cilindro bei plokštumoje.



49 pav. Meteorologinių duomenų klasifikavimas naudojant klasikinę logistinę regresiją

Palyginus duota grafiką su tikrosiomis grupėmis (17 pav.), galima tarti, kad šis modelis pakankamai neblogai gali klasifikuoti turimus duomenis.

Kaip visada, įvertiname klasifikavimo tikslumo matricą be kitus klasifikavimo gerumo matus: tikslumą, specifiškumą, jautrumą bei AUC.

42 lentelė. Meteorologinių duomenų naudojant klasikinę logistinę regresiją klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC
Win	0.889	0.735	0.824	0.867
Spr	0.801	0.668	0.641	0.754
Sum	0.912	0.789	0.860	0.894
Aut	0.812	0.622	0.521	0.712
viso	0.708	0.703	0.712	0.807

Prognozès					
		Win	Spr	Sum	Aut
Grupès	Win	1726	285	0	83
	Spr	441	1681	171	331
	Sum	0	42	1827	255
	Aut	182	510	318	1099

Bendras tokio klasifikavimo tikslumas yra labai panašus, bet vis tiek mažesnis, į cilindrinės logistinės regresijos modelio tikslumą ir lygus 0,708. Klasifikavimo tikslumo matrica parodo, kad dauguma stebinių yra gerai suklasifikuota, tačiau klaidų irgi yra pakankamai nemažai.

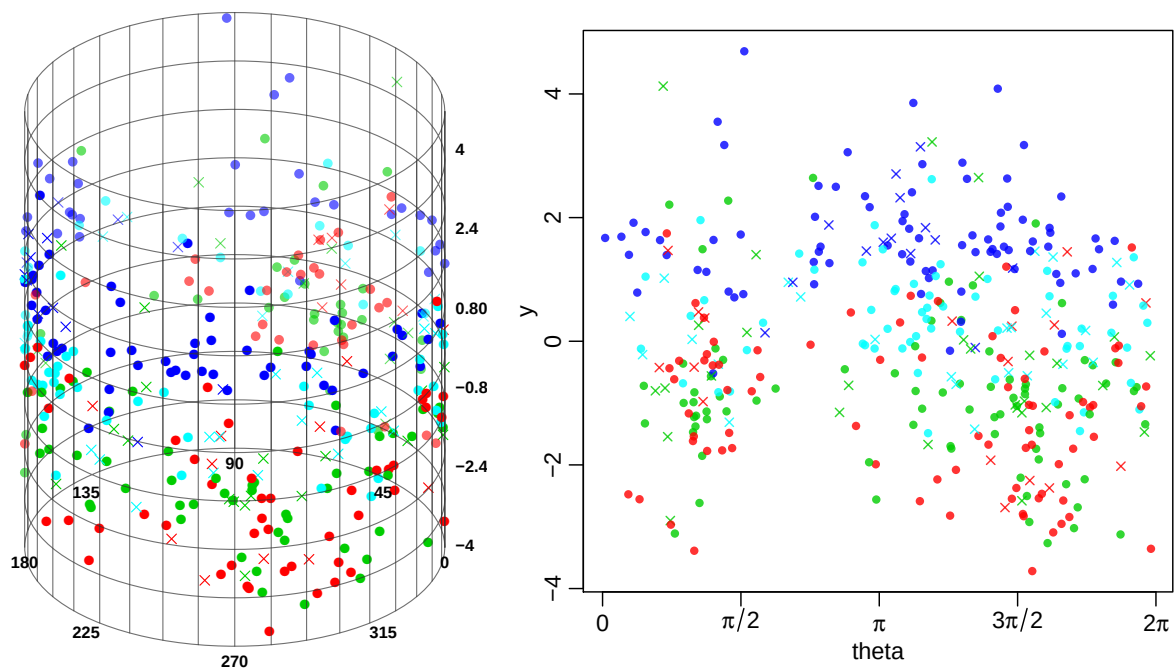
Toliau pritaikome klasikinės logistinės regresijos modelį dieniniams meteorologiniams duomenims. Įvertiname parametrų reikšmingumą ir pateikiame atitinkamas p-reikšmes.

43 lentelė. Klasikinės logistinės regresijos ant dieninių meteorologinių duomenų parametrų reikšmingumo hipotezės p-reikšmės

	Spr	Sum	Aut
(Intercept)	< 1e-16	< 1e-16	< 1e-16
Temp	3.363e-10	< 1e-16	< 1e-16
Hum	5.972e-12	0.0001	0.0224
Precip	0.0117	0.158	0.5662
Wind.dir	0.2427	0.8832	0.883
Wind.spd	0.8425	0.0176	0.359
Press	1.557e-14	< 1e-16	< 1e-16

Šiuo atveju turime vieną nereikšmingą parametą – *Wind.dir*, kuris iš tikro yra skritulinis kintamasis. Toliau pagal nereikšmingumą eina *Wind.spd* ir *Precip*.

Pavaizduojame tokio klasifikavimo rezultatus grafiškai ant cilindro bei plokštumoje.



50 pav. Dieninių meteorologinių duomenų klasifikavimas naudojant klasikinę logistinę regresiją

Palyginus duota grafiką su tikrosiomis grupėmis (20 pav.), galima tarti, kad šis modelis pakankamai neblogai gali klasifikuoti turimus duomenis.

Įvertiname klasifikavimo tikslumo matricą, tikslumą, specifiškumą, jautrumą bei AUC.

44 lentelė. Dieninių meteorologinių duomenų naudojant klasikinę logistinę regresiją klasifikavimo tikslumo matai

	ACC	PREC	REC	AUC	Prognozės				
						Win	Spr	Sum	Aut
Spr	0.844	0.736	0.723	0.808	Grupės	Win	78	7	0
Sum	0.940	0.856	0.902	0.927		Spr	11	81	7
Aut	0.855	0.711	0.648	0.783		Sum	0	3	83
Win	0.925	0.821	0.867	0.905		Aut	6	19	7
viso	0.782	0.781	0.785	0.856				59	

Klasifikavimo matrica parodo, kad daugumoje atvejų priskiriama klasė yra teisinga. Daugiausiai klaidų yra daroma tarp pavasario ir rudens grupių, kaip ir cilindrinės regresijos atveju. Bendras tokio modelio tikslumas šiems duomenims yra 0,782, kas yra truputi blogiau, bet panašu į cilindrinio analogo gautą tikslumą.

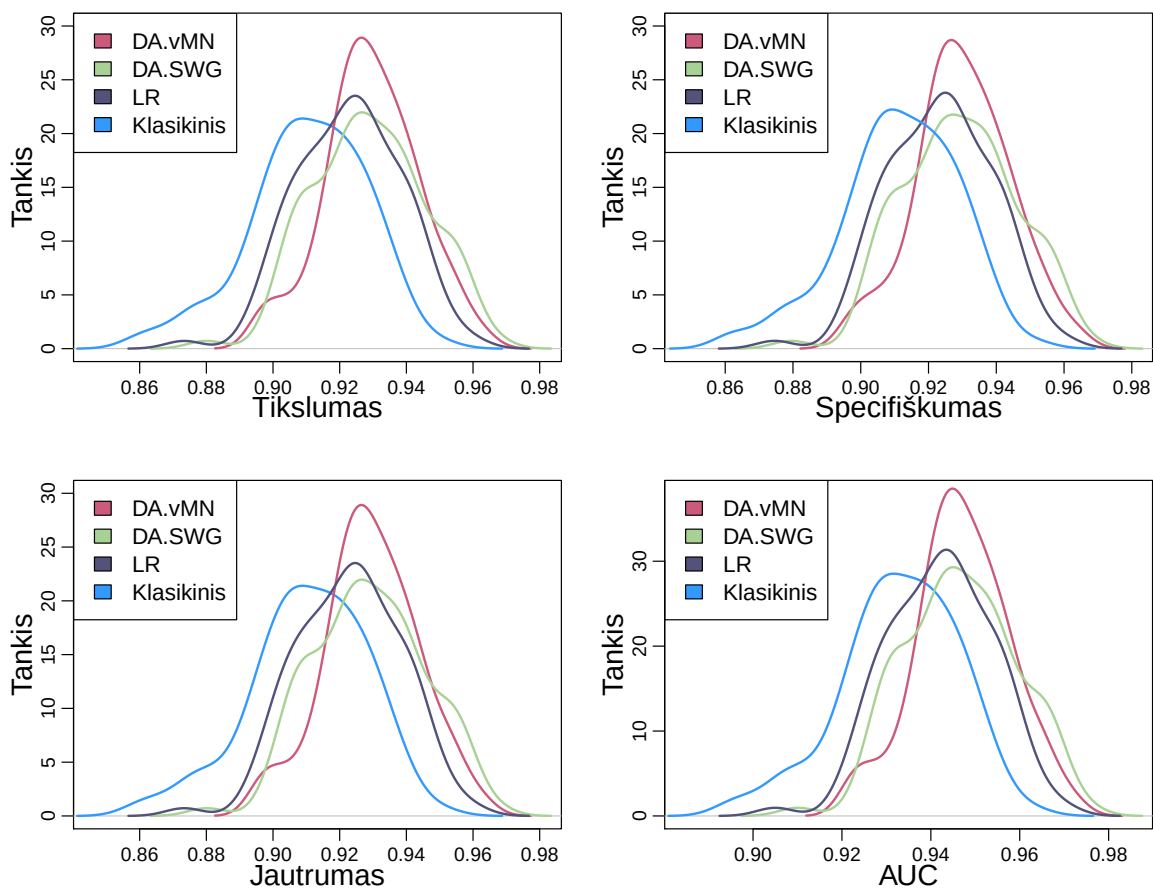
Taigi lyginant klasikinį klasifikavimo būdą su cilindriniais analogais, gavome, kad modelių tikslumas yra gaunamas panašus, bet klasikinio metodo truputi mažesnis. Toks

nedidelis skirtumas tarp šių modelių tikriausiai atsiranda dėl to, kad skritulinis kintamasis labai mažai arba visai neįtakoja klasifikavimo. Tačiau, jei duomenyse yra kryptinis kintamasis, geriau yra naudoti specialius kryptiniams dydžiams skirtus metodus.

2.5.4. Metodų tarpusavyje palyginimas

Šiame skyrelyje skirtingi klasifikavimo ant cilindro metodai yra palyginami tarpusavyje. Tam yra naudojamos duomenų pakartotinis modeliavimas ir jų klasifikavimas, panašiai kaip klasterizavimo atveju. Cilindriniai duomenys yra modeliuojami, taip pat kaip buvo aprašyta skyriuje 2.1.1, 100 kartų. Kiekvienu atveju yra pritaikome keturi klasifikavimo metodai: cilindrinė von Mises-Normalioji diskriminantinė analizė, cilindrinė SWG diskriminantinė analizė, cilindrinė logistinė regresija ir klasikinė logistinė regresija.

Padarius tokias simuliacijas, yra įvertinamas gautų klasių prognozių gerumas. Tam vertinami jau minėti klasifikavimo gerumo matai: tikslumas, specifiškumas, jautrumas ir AUC. Jie yra pateikti tankių grafikuose apačioje.



51 pav. Klasifikavimo ant cilindro metodų charakteristikų tankiai: tikslumas, specifiškumas, jautrumas ir AUC matai

Iš tankių grafikų matome, kad visi klasifikavimo gerumo matai yra žemiausi klasikiniam klasifikavimo metodui, o visų likusių cilindrinų klasifikavimo metodų gerumas yra

panašus.

Tam, kad išrinktume geriausi metodą įvertiname aukščiau pateiktų charakteristikų vidurkius kiekvienam metodui.

45 lentelė. Klasifikavimo ant cilindro metodų vidutinės charakteristikos: tikslumas, specifinis kumas, jautrumas ir AUC matai

	ACC	PREC	REC	AUC
DA.vMN	0.930	0.930	0.930	0.947
DA.SWG	0.929	0.929	0.929	0.947
LR	0.920	0.921	0.920	0.940
Klasikinis	0.911	0.912	0.911	0.933

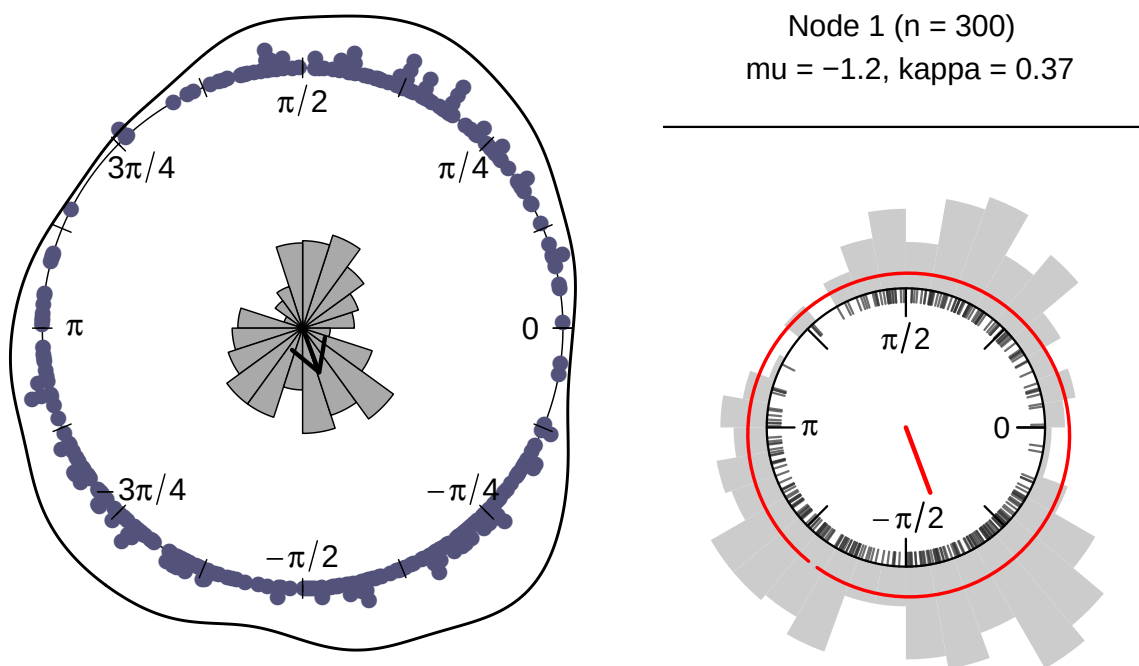
Iš lentelės matome, kad dviejų diskriminantinės analizės metodo cilindriniai analogai turi beveik tokius pačius gerumo matų vidurkius (ir jie yra aukščiausi), t. y. šie metodai yra vienodai geri. Tačiau atsižvelgus į tai, kad SWG diskriminantinės analizės metode kiekvienos grupės pasiskirstymo parametrai yra vertinami iteraciniu būdu ir todėl algoritmas yra lietas, o von Mises-Normalioji diskriminantinė analizė tinka tik paprastiems cilindriniais duomenimis, tinkamiausias metodas turi būti pasirenkamas atsižvelgus į duomenų dimensiją. Cilindrinės logistinės regresijos metodas turi truputi prastesnius klasifikavimo gerumo matus nei cilindriniai diskriminantinės analizės metodai, o klasikinės regresijos metodas, kaip visada, turi prasčiausias charakteristikas.

Vadinasi, jei duomenys turi paprastą cilindrinę struktūrą, tai geriausias metodas duomenų klasifikavimui yra von Mises-Normalioji diskriminantinė analizė.

2.6. Kryptiniai sprendimų medžiai

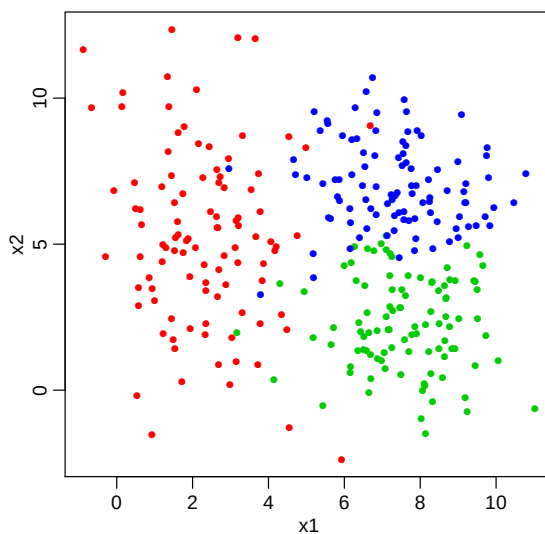
Šiame skyrelyje kryptinius regresijos sprendimų medžius pritaikysime sugeneruotiems bei realiesiems meteorologinėms duomenims.

Visų pirma, modelį taikysime aukščiau sugeneruotų cilindrinų duomenų skrituliniam kintamajam θ . Tokio dydžio skritulinis sklaidos grafikas ir jo bendras didžiausio tikėtinumo (DT) von Mises parinktas pasiskirstymas yra pavaizduoti žemiau (52 pav.). Bendro parinkto von Mises pasiskirstymo log-tikėtinumo funkcijos reikšmė yra -541,4371.



52 pav. Kintamojo θ bendras pasiskirstymas ir DT von Mises pasiskirstymas

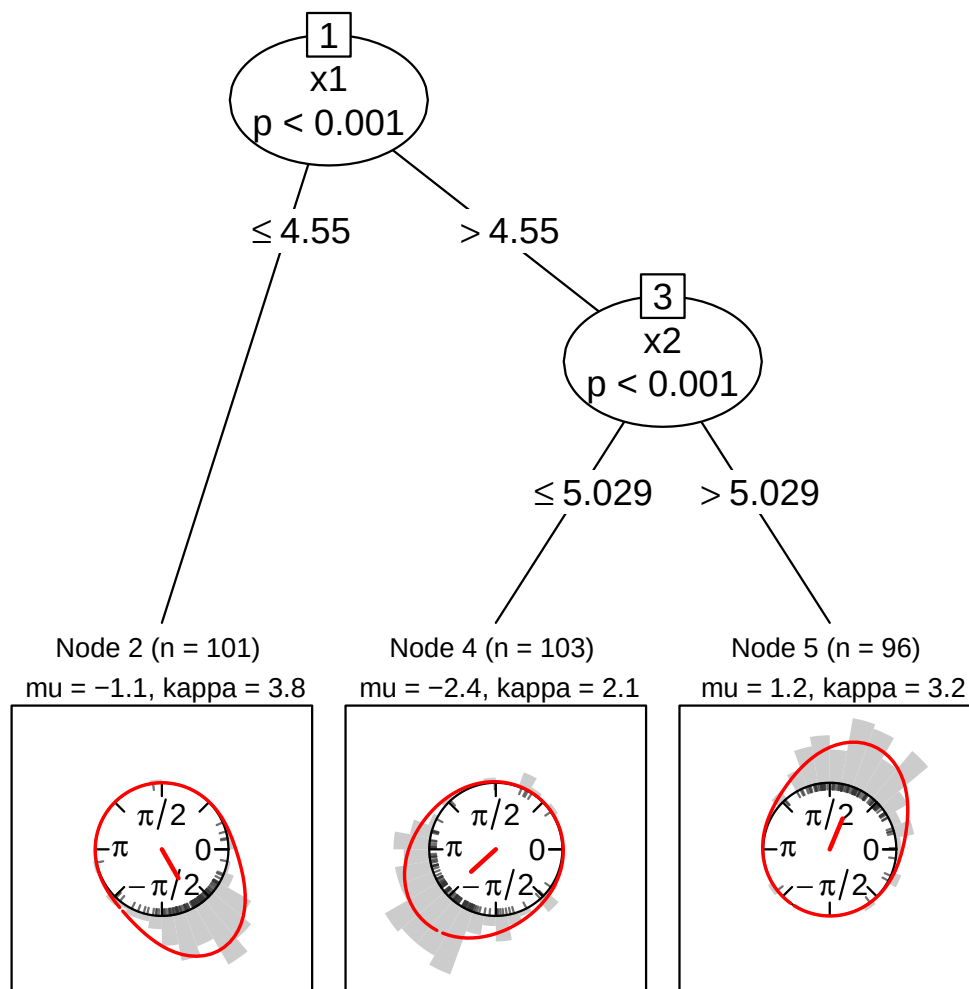
Kadangi bendras kintamojo pasiskirstymas nėra geriausias duomenų modelis (grafikas 52 parodo, kad duomenys yra tri-modalus), toliau pritaikysime kryptinių medžių von Mises modelį, kuris iš pradžių suskaidys duomenis į grupes ir po to kiekvienai grupei suras pasiskirstymą. Tokiam tikslui yra sumodeliuojami du papildomai kintamieji x_1 ir x_2 , pagal kuriuos galima išskirti kintamojo θ 3 grupes. Šių kintamųjų sklaidos diagrama su grupėmis pavaizduota žemiau.



53 pav. Papildomai kintamųjų x_1 ir x_2 , pagal kuriuos įmanoma atskirti kintamojo θ grupes, sklaidos diagrama

Pritaikius kryptinių medžių von Mises modelį naudojant R paketą `circtree` (Lang

et al., 2020b), gaunami sekantys rezultatai:



54 pav. Medžių modelis su sumodeliuotais duomenimis

Matome, kad abu kintamieji x_1 ir x_2 yra reikšmingi modelyje – hipotezės apie kintamojo ir indėlių funkcijos reikšmių nepriklausomumą yra atmetamos, abi p-reikšmės yra mažesnės už 0.001, todėl mažesnės ir už reikšmingumo lygmenį $\alpha = 0,05$. Gruputo pasiskirstymo log-tikėtinumo funkcijos reikšmė yra -303,2288, t. y. ji padidėjo 1,8 karto.

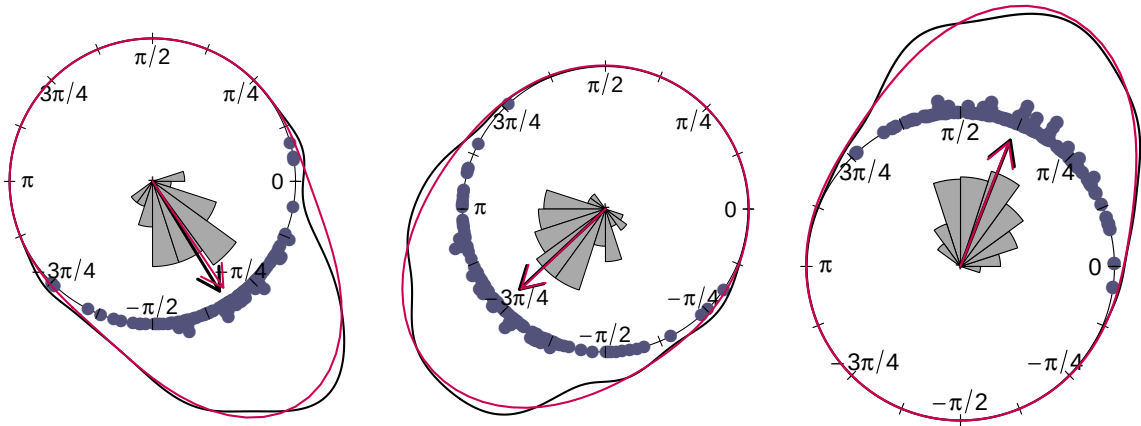
46 lentelė. Atsako kintamojo θ prognozuotų grupių pasiskirstymo parametrai (kairėje), originalių grupių pasiskirstymo parametrai iš lentelės 2 (dešinėje)

	Grupė 1	Grupė 2	Grupė 3		N	μ_1	κ	μ_2	σ
n	101	103	96		100	5.3	5	3	4
μ	5.2	3.9	1.2		100	3.9	3	2.5	2
κ	3.8	2.1	3.2		100	1.2	4	7.5	3

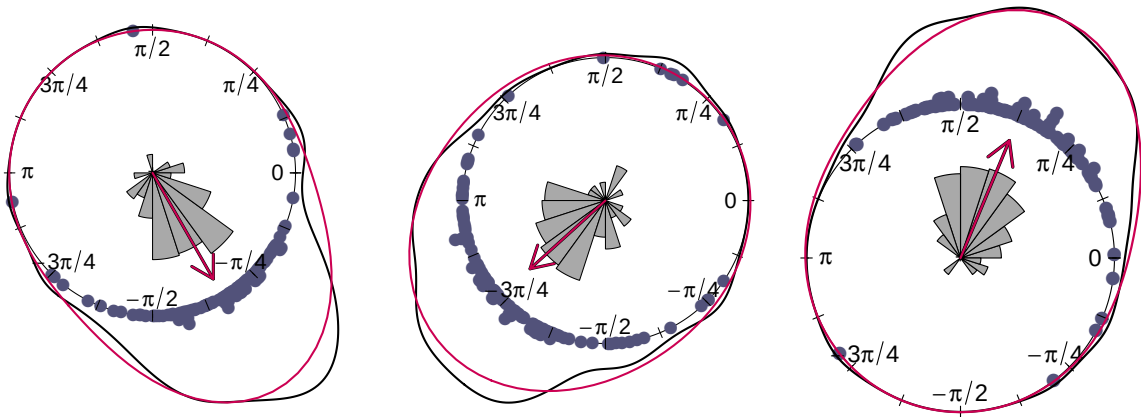
Lyginant gautus rezultatus su tikrais grupių parametrais, matome, kad atspėtos grupės

yra panašios į tikras. Visų grupių dydžiai yra apie 100. Taip pat, įvertintos vidutinės kryptis yra lygios arba apytiksliai lygios tikroms vidutinėms kryptims.

Taip pat palyginimui atvaizduojame tikrų bei suprognuozuotų grupių pasiskirstymus.



55 pav. Tikrų atsako kintamojo θ grupių pasiskirstymai, raudona linija žymi originalų von Mises pasiskirstymo tankį

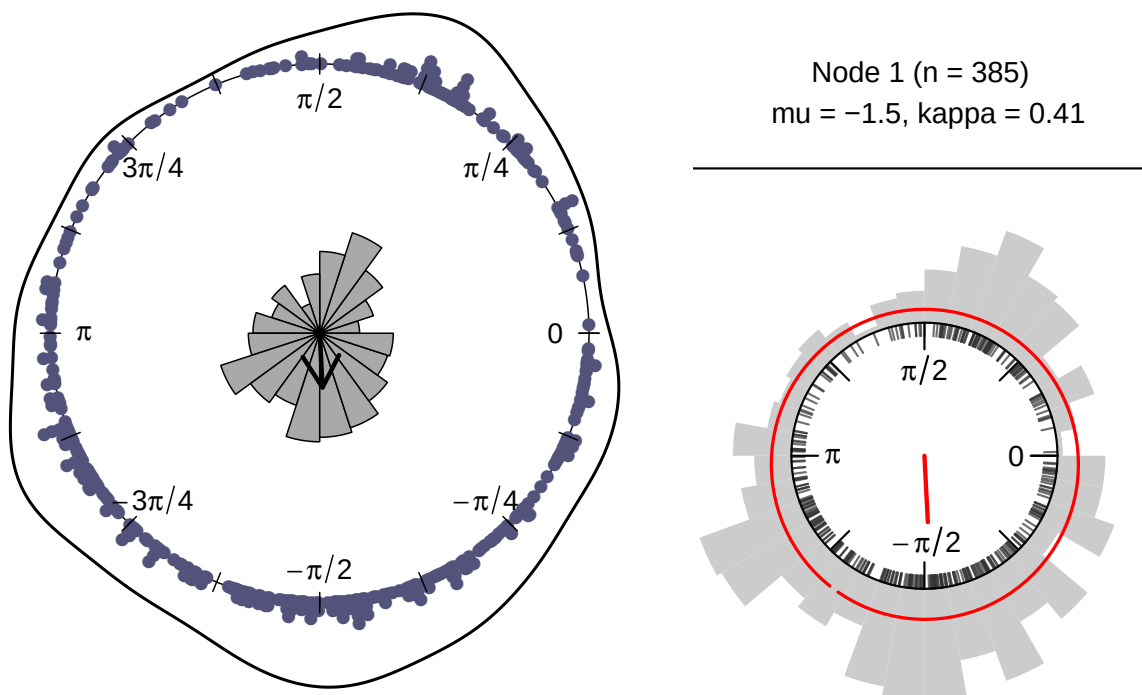


56 pav. Suprognuozuotų atsako kintamojo θ grupių pasiskirstymai, raudona linija žymi parinktą von Mises pasiskirstymo tankį

Lyginant šiuos du grafikus, matome, kad tikri pasiskirstymai ir prognozuoti yra panašūs.

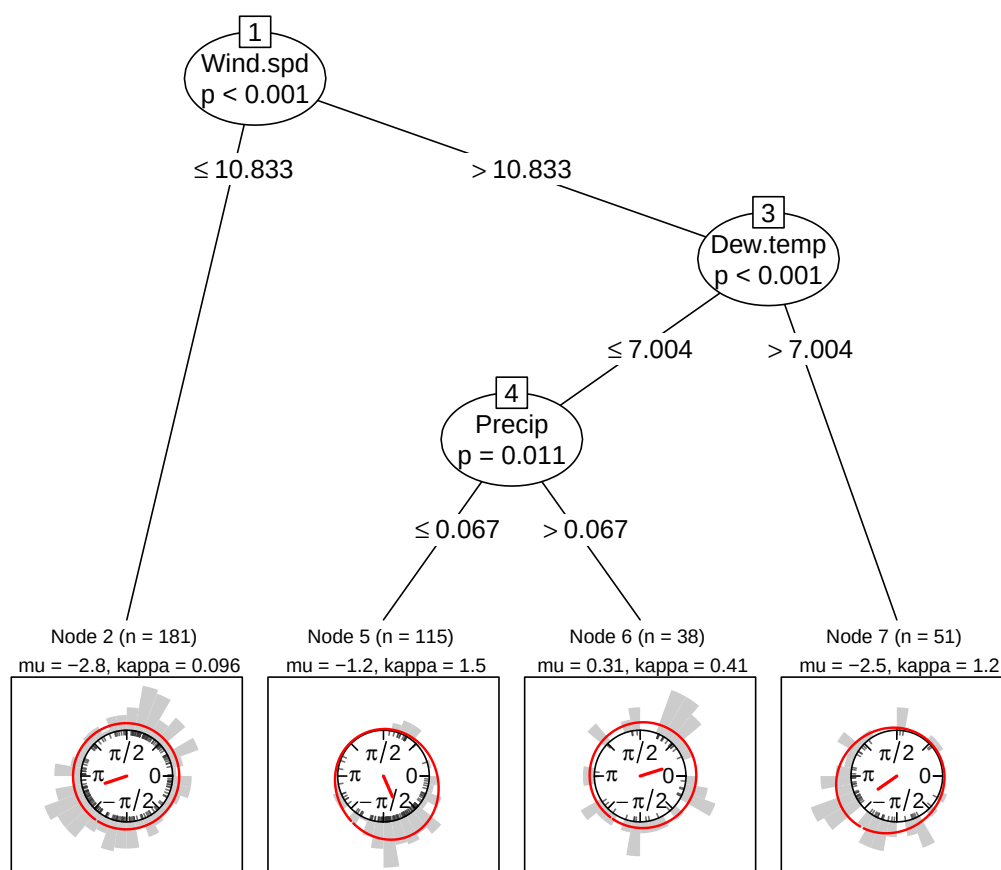
Taigi kryptinių medžių von Mises modelis yra geras įrankis atstatyti nežinomą duomenų pasiskirstymą.

Toliau kryptinių medžių von Mises modelis yra taikomas realiems meteorologinėms duomenims. Rezultatų atvaizdavimo tikslais yra naudojamas suvidurkintų pagal dienas duomenų rinkinys. Modeliavimo kintamasis yra vėjo kryptis. Jo bendras pasiskirstymas ir didžiausio tikėtimumo (DT) von Mises parinktas pasiskirstymas yra pavaizduoti žemiau (57 pav.). Bendro parinkto von Mises pasiskirstymo log-tikėtimumo funkcijos reikšmė yra -691,7448.

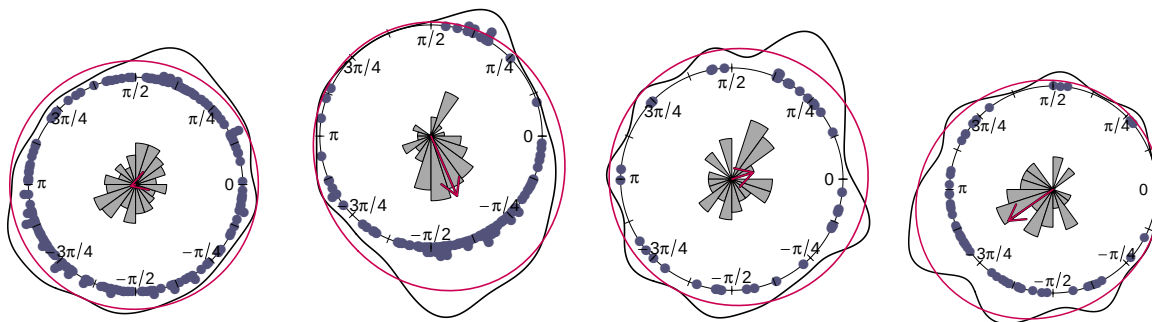


57 pav. Vėjo krypties pasiskirstymas ir DT von Mises pasiskirstymas

Detalesniam vėjo krypties pasiskirstymo radimui naudojame kryptinių medžių von Mises modelį. Šio modelio rezultatas pavaizduotas 58 pav. ir 59 pav.



58 pav. Vėjo krypties medžių modelis



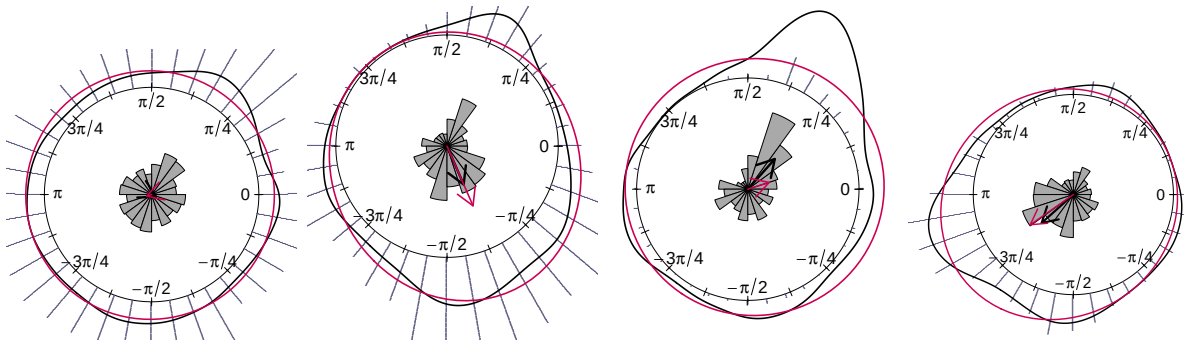
59 pav. Suprognuozuotų vėjo krypčių grupių pasiskirstymai dieniniams duomenims, raudona linija žymi parinktą von Mises pasiskirstymo tankį

Kaip matote vėjo kryptis buvo padalintos į 4 grupes su skirtingais pasiskirstymo parametrais. Tokio pasiskirstymo log-tikėtinumo funkcijos reikšmė padidėjo iki -646,823 (nuo -691,7748). Taip pat be vėjo pasiskirstymo, gauname kintamuosius, nuo kurių labiausiai priklauso vėjo kryptis – vėjo greitis, temperatūra ir kritulių kiekis.

Jei vėjo greitis yra mažas ($< 10,8$), tai vėjo krypties pasiskirstymas yra praktiškai tolygusis. Priešingu atveju, vėjo kryptys yra labiau koncentruotos ties tam tikru kampu ir priklauso nuo kitų meteorologinių parametų:

- jei dirvožemio temperatūra yra aukšta (> 7), tai vėjas pučia $216,76$ laipsniu kampu, t. y. pietvakarių kryptimi.
- jei dirvožemio temperatūra yra žema (< 7) ir beveik nėra kritulių, tai vėjas pučia $291,24$ laipsniu kampu, t. y. šiaurės vakarų kryptimi.
- jei dirvožemio temperatūra yra žema (< 7) ir yra kritulių, tai vėjas pučia $17,76$ laipsniu kampu, t. y. šiaurės kryptimi.

Pritaikome gautą kryptinių sprendimų medžių modelį originaliems, nevidurkintiems, meteorologiniams duomenims. Nubraižome suprognuotų grupių tikrus bei priderintus von Mises pasiskirstymus.



60 pav. Suprognuotų vėjo krypčių grupių pasiskirstymai, raudona linija žymi parinktą von Mises pasiskirstymo tankį

Iš grafiko, matome, kad suskaidymas į grupes yra tinkamas, pirmos grupės pasiskirstymas yra kryptinis tolygusis, o kitų grupių vėjo krypties vidutinės kryptys atitinka prognozes. Tačiau grupių pasiskirstymų koncentracijos parametrai šiek tiek skiriasi nuo padarytų prognozių. Vadinasi naudojant kryptinių sprendimų medžių modelį galima apytiksliai atstatyti vėjo krypties pasiskirstymo dėsnį.

Išvados

Atlikus daugiamačių kryptinių duomenų analizę, gautos išvados:

1. Taikant klasikinius modeliavimo metodus kryptiniams duomenims, gaunami blogesni rezultatai (klasterizavimo apytiksliai 20–35 % blogesni, klasifikavimo – 2–5 %), kartais net beprasmingai, lyginant su jų kryptiniais analogais.
2. Tiek sferiniams, tiek cilindriniais dydžiams k -vidurkių klasterizavimo metodo kryptiniai analogai veikia greičiau (apytiksliai nuo 1.3 iki 4 kartų priklausomai nuo algoritmų) ir taip pat gerai arba geriau (atstumai iki tikrųjų centrų apytiksliai 0–5 % mažesni) nei EM algoritmų kryptiniai analogai.
3. Sferiniams dydžiams logistinės regresijos kryptiniai analogai veikia geriau nei diskriminantinės analizės kryptiniai analogai (bendras klasifikavimo tikslumas apytiksliai 2 % didesnis), o paprastiems cilindriniais dydžiams diskriminantinės analizės analogai yra pranašesni už logistinės regresijos analogus (bendras klasifikavimo tikslumas apytiksliai 1 % didesnis).
4. Kryptinių medžių von Mises modelis yra geras įrankis atstatyti nežinomą skritulinio kintamojo pasiskirstymą, kai jis priklauso nuo papildomų kintamųjų (logtikėtinumai gali padidėti daugiau nei 7 %, priklausomai nuo duomenų).

Literatūra

- Adler, D. & Murdoch, D. (2019). Rgl: 3D Visualization Using OpenGL. <https://CRAN.R-project.org/package=rgl>.
- Al-Daffaie, K. & Khan, S. (2017). Logistic regression for circular data. In *The 3rd ISM International Statistical Conference: Bringing Professionalism and Prestige in Statistics*. doi:10.1063/1.4982860.
- Banerjee, A., Dhillon, I., Ghosh, J. & Sra, S. (2005). Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research* , 1345–1382. doi:10.5555/1046920.1088718.
- Fernandes, K. & Cardoso, J. S. (2016). Discriminative directional classifiers. *Neurocomputing* , 141–149. doi:10.1016/j.neucom.2016.03.076.
- Figueiredo, A. (2009). Discriminant Analysis for the von Mises-Fisher Distribution. *Communications in Statistics - Simulation and Computation*. doi:10.1080/03610910903200281.
- Fujita, K. (2017). A Clustering Method for Data in Cylindrical Coordinates. *Mathematical Problems in Engineering*. doi:10.1155/2017/3696850.
- Hoffleit, D. & Warren, J. W. (1991). The Bright Star Catalog, 5th Revised Edition. <https://heasarc.gsfc.nasa.gov/W3Browse/star-catalog/bsc5p.html>.
- Jammalamadaka, S. R. & Sengupta, A. (2001). *Topics in circular statistics*. Series on Multivariate Analysis. World Scientific.
- Kesemen, O., Tezel, Ö. & Özkul, E. (2016). Fuzzy c-means clustering algorithm for directional data (FCM4DD). *Expert Systems with Applications* , 76–82. doi:10.1016/j.eswa.2016.03.034.
- Lang, M. N., Schlosser, L., Hothorn, T., Mayr, G. J., Stauffer, R. & Zeileis, A. (2020a). Circular Regression Trees and Forests with an Application to Probabilistic Wind Direction Forecasting. doi:10.1111/rssc.12437.
- Lang, M. N., Schlosser, L. & Zeileis, A. (2020b). Circree: Regression trees and forests for circular responses. <https://R-Forge.R-project.org/projects/partykit/>.
- Ligges, U. & Maechler, M. (2003). Scatterplot3d - an R Package for Visualizing Multivariate Data. *Journal of Statistical Software* , 1–20. <http://www.jstatsoft.org>.
- Lund, U. & Agostinelli, C. (2017). R package circular: Circular Statistics. <https://r-forge.r-project.org/projects/circular/>.

- Mardia, K. V. (1972). *Statistics of directional data*. Probability and Mathematical Statistics. Academic Press.
- Mardia, K. V. & Jupp, P. E., eds. (1999). *Directional Statistics*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc. doi:[10.1002/9780470316979](https://doi.org/10.1002/9780470316979).
- Mardia, K. V. & Sutton, T. W. (1978). A Model for Cylindrical Variables with Applications. *Journal of the Royal Statistical Society: Series B (Methodological)* , 229–233. doi:[10.1111/j.2517-6161.1978.tb01668.x](https://doi.org/10.1111/j.2517-6161.1978.tb01668.x).
- Pewsey, A., Neuhäuser, M. & Ruxton, G. D. (2013). *Circular statistics in R*. Oxford University Press.
- R Core Team (2019). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. <https://www.R-project.org/>.
- Roy, A., Parui, S. K. & Roy, U. (2016). SWGMM: A semi-wrapped Gaussian mixture model for clustering of circular–linear data. *Pattern Anal Applic* , 631–645. doi:[10.1007/s10044-014-0418-2](https://doi.org/10.1007/s10044-014-0418-2).
- Soetaert, K. (2017). plot3D: Plotting Multi-Dimensional Data. <https://CRAN.R-project.org/package=plot3D>.
- Tsagris, M. & Alenazi, A. (2019). Comparison of discriminant analysis methods on the sphere. *Communications in Statistics: Case Studies, Data Analysis and Applications* , 467–491. doi:[10.1080/23737484.2019.1684854](https://doi.org/10.1080/23737484.2019.1684854).
- Tsagris, M., Athineou, G., Sajib, A., Amson, E. & Waldstein, M. J. (2020). Directional: Directional statistics. <https://CRAN.R-project.org/package=Directional>.
- von Mises, R. (1918). über die ‘Ganzzahligkeit’ der Atomgewichte und verwandte Fragen. *Physikalische Zeitschrift* , 490–500.

A. Programinis kodas

Apibrėžtos naujos funkcijos.

```
1  # -----
2  # Packages
3  # -----
4  library(scatterplot3d)
5  library(Directional)
6  library(circular)
7  library(circtree)
8  library(lubridate)
9
10 # -----
11 # General functions for angles
12 # -----
13 # Degrees conversion to radians
14 deg.rad <- function(deg){
15   return(deg * pi / 180)
16 }
17 # Radians conversion to degrees
18 rad.deg <- function(rad){
19   return(rad * 180 / pi)
20 }
21 # Check if x is in interval [a, b)
22 between <- function(x, a, b){
23   return((x >=a) & (x < b))
24 }
25 # Change zero direction into b
26 circ.break <- function(theta, b=0){
27   return((theta - b) %% (2*pi) - b)
28 }
29
30 # -----
31 # Functions for Circular Trees
32 # -----
33 c.plot <- function(theta, stack=1, cex=.7, col="#52527a", shrink=1.7,
34                    prop=1.3, main="", text.cex=cex+.15){
35   theta <- circular(na.omit(theta), type="angles", units="radians",
36                    zero=circular(0), rotation="counter")
37   if (main==""){
38     par(mar=c(0,0,0,0))
39   } else {
40     par(mar=c(0, 0, 2, 0))
41   }
42   plot(theta, cex=cex, bin=720, stack=stack, main=main,
43        sep=0.025, shrink=shrink, col=col, axes=FALSE)
44   rose.diag(theta, bins=20, col="darkgrey", cex=cex+.15, ticks=FALSE,
45            prop=prop, add=TRUE, shrink=shrink, axes=FALSE, main=main)
46   lines(density.circular(theta, bw=40), lwd=2, lty=1)
47   axis.circular(at=circular(seq(0, 7*pi/4, pi/4), zero=circular(0),
48                             rotation='counter'),
49               labels=expression(0, pi/4, pi/2, 3*pi/4, pi, -3*pi/4, -pi/2, -pi/4),
50               zero=circular(0), rotation='counter', cex=text.cex)
51   ticks.circular(circular(seq(0, 2*pi, pi/8)), zero=0,
52                 rotation='counter', tcl=0.075)
53   arrows.circular(mean(theta), y=rho.circular(theta), lwd=3)
54 }
```

```

55 tree.gr.plot <- function(theta, mu, kappa, ...){
56   c.plot(theta, ...)
57   curve.circular(dvonmises(x, circular(mu), kappa), add=TRUE, join=TRUE,
58                 lwd=2, zero=circular(0), rotation="counter",
59                 col="#cc0052")
60   arrows.circular(circular(mu), y=A1(kappa), lwd=2, col="#cc0052")
61 }
62
63 # -----
64 # Functions for spherical graphs
65 # -----
66 # Transformation to unambiguous values: phi + 180, -theta
67 norm.coord <- function(theta, phi){
68   phi <- ifelse(theta > pi, (phi + pi) % (2*pi), phi)
69   theta <- ifelse(theta > pi, 2*pi - theta, theta)
70   return(cbind(theta, phi))
71 }
72 # Sphere coordinates from two angles
73 sphere.coord <- function(ang1, ang2){
74   z <- cos(ang1)
75   x <- sin(ang1) * cos(ang2)
76   y <- sin(ang1) * sin(ang2)
77   return(cbind(x, y, z))
78 }
79 # Sphere colors visibility
80 sphere.color <- function(ang1, ang2, col){
81   Col <- ifelse(xor(between(rad.deg(ang2), 10, 190),
82                   between(rad.deg(ang1), 170, 350)),
83               scales::alpha(col, .7), col)
84   return(Col)
85 }
86 # Plot data on the sphere
87 plot.sphere <- function(theta, phi, cex=.7, col="#52527a", pch=16){
88   ang <- seq(0, 2*pi, len=100)
89   A <- do.call(rbind, lapply(seq(0, 2*pi, len=37), sphere.coord, ang1=ang))
90   B <- do.call(rbind, lapply(seq(0, 2*pi, len=37), sphere.coord, ang2=ang))
91   plt <- scatterplot3d(rbind(A, B), angle=60, scale.y=0.4, type="n",
92                       box=0, grid=0, ann=0, axis=0, mar=c(0,0,0,0)+.1)
93   plt$points3d(sphere.coord(theta, phi), pch=pch, cex=cex,
94               col=sphere.color(theta, phi, col))
95   plt$points3d(rbind(A, B), type="l", col=gray(.2, alpha=0.7))
96   lab <- seq(0, 359, 45)
97   text(plt$xyz.convert(sphere.coord(pi/2, deg.rad(lab))), paste(lab),
98       cex=.9, font=2)
99   text(plt$xyz.convert(sphere.coord(deg.rad(lab)[1:5], 0)), paste(lab[1:5]),
100      cex=.9, font=2, pos=4)
101   return(plt)
102 }
103 # Plot spherical data normally
104 plot.normal.sphere <- function(theta, phi, col="#52527a", pch=16){
105   X <- norm.coord(theta, phi)
106   theta <- X[,1]
107   phi <- X[,2]
108   plot.default(phi, theta, col=scales::alpha(col, 0.8), pch=pch, cex=.8,
109               xaxt="none", yaxt="none")
110   axis(side=1, at=seq(0, 2*pi, pi/2),
111       labels=expression(0, pi/2, pi, 3*pi/2, 2*pi))
112   axis(side=2, at=seq(0, pi, pi/4),
113       labels=expression(0, pi/4, pi/2, 3*pi/4, pi))
114 }

```

```

115
116 # -----
117 # Function for spherical analysis
118 # -----
119 # Spherical mean vector
120 sphere.mean <- function(coord){
121   if (ncol(coord)==2) coord <- sphere.coord(coord[,1], coord[,2])
122   x <- colMeans(coord)
123   R <- norm(x, "2")
124   x0 <- x / R
125   return(c(x0, R))
126 }
127 # Get two angles from sphere coordinates
128 coord2ang <- function(X){
129   theta <- acos(X[,3]) # [0, pi]
130   phi <- ifelse(round(sin(theta), 6)==0,
131                 0, atan2(X[,2], X[,1]) %%(2*pi)) # [0, 2pi]
132   cbind(theta, phi)
133 }
134 # Distance between points on sphere
135 sphere.dist <- function(theta, phi, m){
136   X <- sphere.coord(theta, phi)
137   return(acos(tcrossprod(m, X))[1,])
138 }
139 # Spherical k-means clustering
140 sphere.kmeans <- function(theta, phi, k, max.iter=50, tol=.0){
141   data <- cbind(theta, phi)
142   group <- list(sample(1:k, nrow(data), TRUE))
143   for (i in 1:max.iter){
144     X0 <- by(data, group[[i]], sphere.mean)
145     Cmu <- matrix(unlist(X0), ncol=4, byrow=TRUE)[-4]
146     D <- apply(Cmu, 1, sphere.dist, theta=theta, phi=phi)
147     group[[i+1]] <- apply(D, 1, which.min)
148     if(sum(group[[i]] !=group[[i+1]]) <=tol*nrow(data)) break
149   }
150   return(list(pred=group[[length(group)]], iter=i,
151              error=sum(group[[i]] !=group[[i+1]])))
152 }
153
154 # Classification goodness of fit
155 confusion_matrix <- function(pred, ytest){
156   cm <- table(ytest, pred)
157   image(cm, axes=0, xlab=iconv('Grupès', "UTF-8", "UTF-8"),
158         ylab=iconv('Proгнозès', "UTF-8", "UTF-8"))
159   x <- seq(0, 1, len=nrow(cm))
160   y <- seq(0, 1, len=ncol(cm))
161   axis(1, x, rownames(cm), tick=FALSE, font=2, cex=1.2)
162   axis(2, y, colnames(cm), tick=FALSE, font=2, cex=1.2)
163   text(rep(x, nrow(cm)), rep(y, each=ncol(cm)), cm, cex=2)
164   return(cm)
165 }
166 library(Metrics)
167 metrics <- function(pred, ytest){
168   cl <- unique(ytest)
169   acc <- sapply(cl, function(k) accuracy(ytest==k, pred==k))
170   prec <- sapply(cl, function(k) precision(ytest==k, pred==k))
171   rec <- sapply(cl, function(k) recall(ytest==k, pred==k))
172   auc <- sapply(cl, function(k) auc(ytest==k, pred==k))
173   M <- cbind(acc, prec, rec, auc)
174   rownames(M) <- cl

```

```

175   avg.M <- c(acc=accuracy(ytest, pred), colMeans(M[,2:4]))
176   return(rbind(M, avg=avg.M))
177 }
178
179 # -----
180 # Functions for cylindrical graphs
181 # -----
182 # transform angle + variable into cylinder coordinates
183 cylinder.coord <- function(theta, y){
184   x1 <- cos(theta)
185   x2 <- sin(theta)
186   return(cbind(x1, x2, y))
187 }
188 # define color visibility on cylinder
189 cylinder.color <- function(theta, col){
190   Col <- ifelse(theta < pi, scales::alpha(col, .6), col)
191   return(Col)
192 }
193 # plot data points on cylinder
194 plot.cylinder <- function(theta, y, col="#52527a", cex=.8, pch=16){
195   n <- 10
196   by <- signif(diff(range(y)) / n, 1)
197   start <- by * floor(min(y) / by)
198   end <- by * ceiling(max(y) / by)
199   A <- do.call(rbind, lapply(seq(start, end, len=n), cylinder.coord,
200     theta=seq(0, 2*pi, len=100)))
201   plt <- scatterplot3d(A, angle=90, scale.y=.5, type="n", box=0,
202     grid=0, ann=0, axis=0, mar=c(0,0,0,0)+.1,
203     zlim=c(start, end))
204   plt$points3d(cylinder.coord(theta, y), pch=pch, cex=cex,
205     col=cylinder.color(theta, col))
206   plt$points3d(A, type="l", col=gray(.2, alpha=0.7))
207   . <- sapply(seq(0, 2*pi, len=37), function(x) plt$points3d(
208     cylinder.coord(x, c(end, start)), type="l",
209     col=gray(.2, alpha=0.7)))
210   lab1 <- seq(0, 359, 45)
211   lab2 <- seq(start, end, by=2*by)
212   text(plt$xyz.convert(cylinder.coord(deg.rad(lab1), start)), paste(lab1),
213     cex=.9, font=2, pos=1)
214   text(plt$xyz.convert(cylinder.coord(0, lab2)), paste(lab2),
215     cex=.9, font=2, pos=4)
216   return(plt)
217 }
218 # plot cut on zero cylinder data
219 plot.normal.cylinder <- function(theta, y, col="#52527a", cex=.8, pch=16){
220   plot.default(theta, y, col=scales::alpha(col, 0.8), pch=pch, cex=cex,
221     xaxt="none")
222   axis(side=1, at=seq(0, 2*pi, pi/2),
223     labels=expression(0, pi/2, pi, 3*pi/2, 2*pi))
224 }
225
226 # -----
227 # Functions for cylindrical analysis
228 # -----
229 # Mean vector on cylinder
230 cyl.mean <- function(theta, y=NULL){
231   if (is.null(y)){
232     X <- theta
233   } else {
234     X <- cbind(theta, y)

```

```

235 }
236 m <- colMeans(X)
237 m[1] <- mean(circular(X[,1]))
238 return(m)
239 }
240 # Distance on cylinder
241 cyl.dist1 <- function(t1, y1, t2, y2, kappa=1, sigma=sd(y1)){
242   d <- kappa * (1 - cos(t1 - t2)) + sqrt((y1 - y2)^2) / sigma
243   return(d)
244 }
245 cyl.dist2 <- function(t1, y1, t2, y2, kappa=1, sigma=sd(y1)){
246   d <- kappa * cos(t1 - t2) - (y1 - y2)^2 / sigma^2
247   return(-d)
248 }
249 # cyk-means (cylinder k-means)
250 cyk.means <- function(theta, y, k, fixed=TRUE, max.iter=50, tol=.0){
251   if (fixed){
252     sigma_fun <- function(y, gr=NULL, k=NULL) sqrt(2)*sd(y)
253     kappa_fun <- function(t, gr=NULL, k=NULL) 1
254     cyl.dist <- cyl.dist1
255   } else {
256     sigma_fun <- function(y, gr, k) sqrt(2)*sd(y[gr==k])
257     kappa_fun <- function(t, gr, k) A1inv(rho.circular(circular(t[gr==k])))
258     cyl.dist <- cyl.dist2
259   }
260   data <- cbind(theta, y)
261   group <- list(sample(1:k, nrow(data), TRUE))
262   for (i in 1:max.iter){
263     gr <- group[[i]]
264     M <- by(data, gr, cyl.mean)
265     D <- sapply(1:length(unique(gr)), function(j)
266       cyl.dist(theta, y, M[[j]][1], M[[j]][-1],
267         kappa_fun(theta, gr, j), sigma_fun(y, gr, j)))
268     group[[i+1]] <- apply(D, 1, which.min)
269     if(sum(group[[i]] !=group[[i+1]]) <=tol*nrow(data)) break
270   }
271   gr <- group[[length(group)]]
272   return(list(pred=gr, centers=by(data, gr, cyl.mean),
273     iter=i, error=sum(group[[i]] !=group[[i+1]])))
274 }
275
276 # Fit vonMises-Normal cylindrical distribution
277 fit.vMN <- function(theta, y, weights=NULL){
278   if (is.null(weights)){
279     # Circular
280     mu0 <- as.numeric(circ.break(mean(circular(theta))))
281     R <- rho.circular(circular(theta))
282     kappa <- A1inv(R)
283     # Conditional linear
284     data <- data.frame(y=y, C=cos(theta), S=sin(theta))
285     mu.theta <- lm(y ~ 1 + C + S, data=data)
286     sigma.c <- sd(resid(mu.theta))
287     sigma <- sd(y)
288   } else {
289     # Circular
290     C <- crossprod(weights, cos(theta)) / sum(weights)
291     S <- crossprod(weights, sin(theta)) / sum(weights)
292     mu0 <- drop(circ.break(atan2(S, C)))
293     R <- drop(sqrt(C^2 + S^2))
294     kappa <- A1inv(R)

```

```

295 # Conditional linear
296 data <- data.frame(y=y, C=cos(theta), S=sin(theta))
297 mu.theta <- lm(y ~ 1 + C + S, data=data, weights=weights)
298 e <- resid(mu.theta)
299 sigma.c <- sqrt(drop(crossprod(weights, e^2) / sum(weights)))
300 m <- drop(crossprod(weights, y) / sum(weights))
301 sigma <- sqrt(drop(crossprod(weights, (y - m)^2) / (sum(weights) - 1)))
302 }
303 # Parameters for conditional linear
304 B <- coef(mu.theta)
305 mu <- as.numeric(B[1])
306 rho <- sqrt(1 - sigma.c^2 / sigma^2)
307 nu <- circ.break(as.numeric(atan2(B[3], B[2])))
308 param <- c(mu0=mu0, kappa=kappa, mu=mu, sigma=sigma, rho=rho, nu=nu)
309 mlist <- list(mu0=mu0, kappa=kappa, mu.theta=mu.theta, sigma.c=sigma.c)
310 return(list(param=param, mlist=mlist))
311 }
312 # Estimate vonMises-Normal cylindrical density
313 dvMN <- function(theta, y, mu0, kappa, mu.theta, sigma.c){
314   newD <- data.frame(C=cos(theta), S=sin(theta))
315   mu <- predict(mu.theta, newD)
316   N <- dnorm(y, mu, sigma.c)
317   vM <- circular::dvonmises(circular(theta), circular(mu0), kappa)
318   return(N * vM)
319 }
320 # EM vonMises-Normal clustering on cylinder
321 EMvMN <- function(theta, y, K, tol=1e-4, max.iter=50, initKM=TRUE,
322   trace=TRUE){
323   # initial groups
324   N <- length(theta)
325   if (initKM){
326     g <- cyk.means(theta, y, K, fixed=TRUE)$pred
327   } else {
328     g <- sample(1:K, N, TRUE)
329   }
330   Gamma.i <- function(x){
331     d <- sapply(1:K, function(k)
332       do.call(dvMN, c(list(theta=x[1], y=x[2]), param.k[[k]])))
333     pi.k.N <- pi.k * d
334     return(pi.k.N / sum(pi.k.N))
335   }
336   L.i <- function(x){
337     d <- sapply(1:K, function(k)
338       do.call(dvMN, c(list(theta=x[1], y=x[2]), param.k[[k]])))
339     return(log(crossprod(pi.k, d)))
340   }
341   pi.k <- prop.table(table(g))
342   param.k <- lapply(1:K, function(k) fit.vMN(theta[g==k], y[g==k])$mlist)
343   l <- sum(apply(cbind(theta, y), 1, L.i))
344   for (iter in 1:max.iter){
345     # E step
346     gamma <- t(apply(cbind(theta, y), 1, Gamma.i))
347     # M step
348     pi.k <- colSums(gamma) / N
349     param.k <- lapply(1:K, function(k) fit.vMN(theta, y, gamma[,k])$mlist)
350     # convergence
351     l[iter+1] <- sum(apply(cbind(theta, y), 1, L.i))
352     if(trace){
353       cat('Iter=', iter, '   LogL:', signif(l[iter+1], 6), '\n')
354       if(l[iter+1] - l[iter] < tol){

```

```

355     print("Converged")
356     break
357 }
358 } else {
359     if(l[iter+1] - l[iter] < tol){
360         break
361     }
362 }
363 }
364 g <- apply(gamma, 1, which.max)
365 param <- sapply(1:K, function(k) fit.vMN(theta[g==k], y[g==k])$param)
366 out <- list(groups=g, logL=signif(l[iter+1], 6),
367             Iter=iter, param=param, pi.k=pi.k)
368 return(out)
369 }
370
371 # Estimate Semi-wrapped Gaussian cylindrical density
372 dSWG <- function(theta, y, mu, sigma, w=-1:1){
373     d <- sum(sapply(w, function(wi) dmvnorm(c(theta + 2*wi*pi, y), mu, sigma)))
374     return(d)
375 }
376 # Fit SWG distribution model
377 fit.SWG <- function(theta, y, w=-1:1, tol=1e-4, max.iter=50, trace=TRUE){
378     N <- length(theta)
379     wn <- length(w)
380     sigma.num <- function(j){
381         X <- as.matrix(cbind(theta + 2*w[j]*pi, y))
382         s <- lapply(1:nrow(X), function(i) gamma[i,j] * tcrossprod(X[i,] - mu))
383         Reduce('+', s)
384     }
385     Gamma.i <- function(x){
386         d <- sapply(w, function(wi) dSWG(x[1], x[-1], mu, sigma, wi))
387         return(d / sum(d))
388     }
389     # initial parameters
390     mu <- cyl.mean(theta, y)
391     w0 <- sapply(theta, function(t) which.min(t + 2*pi*w - mu[1])^2)
392     gamma <- sapply(1:wn, function(j) w0==j) * 1
393     sigma <- Reduce('+', lapply(1:wn, sigma.num)) / N
394     l <- sum(apply(cbind(theta, y), 1, function(x)
395                 dSWG(x[1], x[-1], mu, sigma)))
396     # MLE estimates
397     for (iter in 1:max.iter){
398         gamma <- t(apply(cbind(theta, y), 1, Gamma.i))
399         mu <- rowSums(sapply(1:wn, function(j)
400             crossprod(gamma[,j], as.matrix(cbind(theta + 2*w[j]*pi, y))))) / N
401         sigma <- Reduce('+', lapply(1:wn, sigma.num)) / N
402         # convergence
403         l[iter+1] <- sum(apply(cbind(theta, y), 1, function(x)
404                             dSWG(x[1], x[-1], mu, sigma)))
405         if(l[iter+1] - l[iter] < tol){
406             if(trace){
407                 cat('(Converged) Iter=', iter, '    LogL:', signif(l[iter+1], 6), '\n')
408             }
409             break
410         }
411     }
412     return(list(mu=mu, sigma=sigma))
413 }
414 # EM SWG (Semi-wrapped Gaussian) clustering on cylinder

```

```

415 EM.SWG <- function(theta, y, K, w=-1:1, tol=1e-4, max.iter=50, initKM=TRUE,
416                      trace=TRUE){
417   # initial groups
418   N <- length(theta)
419   wn <- length(w)
420   if (initKM){
421     g <- cyk.means(theta, y, K, fixed=TRUE)$pred
422   } else {
423     g <- sample(1:K, N, TRUE)
424   }
425   Gamma.i <- function(x, j){
426     d <- sapply(1:K, function(k)
427                 do.call(dSWG, c(list(theta=x[1], y=x[-1]), param[[k]])))
428     den <- sum(pi.k * d)
429     d <- sapply(w, function(wi)
430                 do.call(dSWG, c(list(theta=x[1], y=x[-1], w=wi), param[[j]])))
431     return(pi.k[j] * d / den)
432   }
433   L.i <- function(x){
434     d <- sapply(1:K, function(k)
435                 do.call(dSWG, c(list(theta=x[1], y=x[-1]), param[[k]])))
436     return(log(crossprod(pi.k, d)))
437   }
438   sigma.num <- function(gw, mu, j){
439     X <- cbind(theta + 2*w[j]*pi, y)
440     s <- lapply(1:nrow(X), function(i) gw[i,j] * tcrossprod(X[i,] - mu))
441     Reduce('+', s)
442   }
443   Param.k <- function(gw, wn=wn){
444     den <- sum(gw)
445     mu <- rowSums(sapply(1:wn, function(j)
446                         crossprod(gw[,j], cbind(theta + 2*w[j]*pi, y)))) / den
447     sigma <- Reduce('+', lapply(1:wn, sigma.num, gw=gw, mu=mu)) / den
448     return(list(mu=mu, sigma=sigma))
449   }
450   pi.k <- prop.table(table(g))
451   param <- lapply(1:K, function(k) fit.SWG(theta[g==k], y[g==k], trace=trace))
452   l <- sum(apply(cbind(theta, y), 1, L.i))
453   if(trace){
454     cat('Initial   LogL:', signif(l, 6), '\n')
455     for (iter in 1:max.iter){
456       # E step
457       gamma.w <- lapply(1:K, function(j)
458                           t(apply(cbind(theta, y), 1, Gamma.i, j=j)))
459       gamma <- Reduce(cbind, lapply(gamma.w, rowSums))
460       # M step
461       pi.k <- colSums(gamma) / N
462       param <- lapply(gamma.w, Param.k, wn=wn)
463       # convergence
464       l[iter+1] <- sum(apply(cbind(theta, y), 1, L.i))
465       cat('Iter=', iter, '   LogL:', signif(l[iter+1], 6), '\n')
466       if(l[iter+1] - l[iter] < tol){
467         print("Converged")
468         break
469       }
470     }
471   } else {
472     for (iter in 1:max.iter){
473       # E step
474       gamma.w <- lapply(1:K, function(j)

```

```

475         t(apply(cbind(theta, y), 1, Gamma.i, j=j)))
476     gamma <- Reduce(cbind, lapply(gamma.w, rowSums))
477     # M step
478     pi.k <- colSums(gamma) / N
479     param <- lapply(gamma.w, Param.k, wn=wn)
480     # convergence
481     l[iter+1] <- sum(apply(cbind(theta, y), 1, L.i))
482     if(l[iter+1] - l[iter] < tol){
483         break
484     }
485 }
486 }
487 g <- apply(gamma, 1, which.max)
488 out <- list(groups=g, logL=signif(l[iter+1], 6),
489             Iter=iter, param=param, pi.k=pi.k)
490 return(out)
491 }
492
493 # Cylindrical DA based on vonMises-Normal
494 DA.vMNcyl <- function(theta, y, group){
495     group <- as.factor(group)
496     cl <- unique(group)
497     K <- length(cl)
498     g <- as.numeric(group)
499     pi.k <- prop.table(table(g))
500     param.k <- lapply(1:K, function(k) fit.vMN(theta[g==k], y[g==k])$mlist)
501     d <- sapply(1:K, function(k)
502                 do.call(dvMN, c(list(theta=theta, y=y), param.k[[k]])))
503     delta <- log(d) + matrix(log(pi.k), nrow(d), K, TRUE)
504     pred <- cl[apply(delta, 1, which.max)]
505     return(pred)
506 }
507
508 # Cylindrical DA based on SWG
509 DA.SWGcyl <- function(theta, y, group, ...){
510     y <- as.data.frame(y)
511     group <- as.factor(group)
512     cl <- unique(group)
513     K <- length(cl)
514     g <- as.numeric(group)
515     pi.k <- prop.table(table(g))
516     param.k <- lapply(1:K, function(k) fit.SWG(theta[g==k], y[g==k], ...))
517     d <- sapply(1:K, function(k) apply(cbind(theta, y), 1, function(x)
518                                     do.call(dSWG, c(list(theta=x[1], y=x[-1]), param.k[[k]]))))
519     delta <- log(d) + matrix(log(pi.k), nrow(d), K, TRUE)
520     pred <- cl[apply(delta, 1, which.max)]
521     return(pred)
522 }

```

Duomenų sutvarkymo kodo dalis.

```

1 # -----
2 # Stars Data
3 # -----
4 file <- "data/stars/bsc5.dat"
5 info <- "data/stars/bsc5.readme.txt"
6

```

```

7  # Read dataset description data
8  d <- read.fwf(info, c(4, -1, 3, -6, -10, 10), skip=81, n=53,
9      col.names=c("start", "end", "name"))
10 d$name <- trimws(d$name)
11 d$start <- ifelse(is.na(d$s), d$e, d$s)
12 d$skip <- 1
13 d[c(2:6, 8, 10:12, 30:31, 33, 35:37, 39, 42:43, 45:53), "skip"] <- -1
14 d$w <- (d$end - d$start + 1) * d$skip
15
16 # Import dataset
17 D <- read.fwf(file, d$w, col.names=d[d$skip == 1, 'name'])
18 csvfile <- "data/stars/data.csv"
19 levels(D$IRflag)[1] <- '-'
20 levels(D$Multiple)[1] <- '-'
21 write.csv(D, csvfile, row.names=FALSE)
22
23 # Data Summary
24 csvfile <- "data/stars/data.csv"
25 D <- read.csv(csvfile)
26 dim(D)
27 D <- na.omit(D)
28 dim(D)
29 summary(D)
30 data <- subset(df, select=-c(HR, GLAT, GLON, Multiple, SpType, UB))[,-(2:8)]
31 corrplot::corrplot(cor(data[,!sapply(data, is.factor)]),
32     type = "upper", method = "color", addCoef.col = "black")
33
34 # -----
35 # Meteorology Data
36 # -----
37 # Import one meteo-data file
38 read.meteo.data <- function(f){
39     d <- read.csv(f)[c(5, seq(10, 20, 2), 24)]
40     cols <- c('Date', 'Temp', 'Dew.temp', 'Hum', 'Precip',
41         'Wind.dir', 'Wind.spd', 'Press')
42     names(d) <- cols
43     return(d)
44 }
45 # Import and concatenate all meteo-data files
46 path <- "data/meteo"
47 files <- paste(path, list.files(path, "*H.csv"), sep='/')
48 df <- Reduce(rbind, lapply(files, read.meteo.data))
49 df$Wind.dir <- deg.rad(df$Wind.dir * 10)
50 csvfile <- "data/meteo/data.csv"
51 df <- df[complete.cases(df[-6]),]
52 write.csv(df, csvfile, row.names=FALSE)
53
54 # Data preparation and summary
55 csvfile <- "data/meteo/data.csv"
56 df <- read.csv(csvfile)
57 df$Date <- ymd_hm(df$Date)
58 dim(df)
59 summary(df)
60 corrplot::corrplot(cor(df[-1], use='pairwise.complete.obs'), type = "upper",
61     method = "color", addCoef.col = "black")
62
63 # Transformation to mean daily data
64 date <- floor_date(df$Date, 'day')
65 dfm <- aggregate(. ~ date, df[-1], mean)
66 dfm$Wind.dir <- tapply(df$Wind.dir, date, function(x)

```

```

67         mean(na.omit(circular(x))))
68 dfm$Wind.dir <- circ.break(dfm$Wind.dir)
69 csvfile2 <- "data/meteo/mean_data.csv"
70 write.csv(dfm, csvfile2, row.names=FALSE)
71 dim(dfm)
72 summary(dfm)

```

Pagrindinė kodo dalis.

```

1  # -----
2  # Generate spherical data
3  # -----
4  # generate data with 3 groups
5  set.seed(928)
6  n <- 100
7  theta <- c(rvonmises(n, circular(1.5), 10),
8             rvonmises(n, circular(0), 3),
9             rvonmises(n, circular(1.8), 8))
10 phi <- c(rvonmises(n, circular(4.5), 25),
11          rvonmises(n, circular(3), 5),
12          rvonmises(n, circular(5.2), 8))
13 plt <- plot.sphere(theta, phi, cex=1.3, col=rep(2:4, each=n))
14 plot.normal.shere(theta, phi, col=rep(2:4, each=n))
15
16 # -----
17 # EM Clustering (Spherical)
18 # -----
19 res <- mixvmf.mle(sphere.coord(theta, phi), 3, n.start=10)
20 mu <- res$param[,1:3]
21 param <- cbind(N=table(res$pred),
22               round(cbind(coord2ang(mu), res$param[,4:5]), 3))
23 res$iter
24 param
25
26 plt <- plot.sphere(theta, phi, cex=1.3, col=res$pred+1)
27 plt$points3d(mu, pch='x', cex=2, col=2:4)
28
29 plot.normal.shere(theta, phi, col=scales::alpha(res$pred+1, .7))
30 points(param[,3], param[,2], col=2:4, pch='x', cex=1.5)
31
32 # -----
33 # K-means Clustering on Sphere
34 # -----
35 res <- sphere.kmeans(theta, phi, 3)
36 X0 <- by(sphere.coord(theta, phi), res$pred, sphere.mean)
37 mu <- matrix(unlist(X0), ncol=4, byrow=TRUE)[-4]
38 param <- cbind(N=table(res$pred),
39               round(cbind(coord2ang(mu), res$param[,4:5]), 3))
40 res$iter
41 param
42
43 plt <- plot.sphere(theta, phi, cex=1.3, col=res$pred+1)
44 plt$points3d(mu, pch='x', cex=2, col=2:4)
45
46 plot.normal.shere(theta, phi, col=scales::alpha(res$pred+1, .7))
47 points(param[,3], param[,2], col=2:4, pch='x', cex=1.3)
48

```

```

49 # -----
50 # Linear Clustering (on sphere)
51 # -----
52 X <- norm.coord(theta, phi)
53 res <- kmeans(X, 3, iter.max=50)
54 param <- cbind(N=table(res$cluster), round(res$centers, 3))
55 res$iter
56 param
57
58 plt <- plot.sphere(theta, phi, cex=1.3, col=res$cluster+1)
59 mu <- sphere.coord(res$centers[,1], res$centers[,2])
60 plt$points3d(mu, pch='x', cex=2, col=2:4)
61
62 plot.normal.shere(theta, phi, col=scales::alpha(res$cluster+1, .7))
63 points(res$centers[,2], res$centers[,1], col=2:4, pch='x', cex=1.3)
64
65 # -----
66 # Real Spherical Data
67 # -----
68 csvfile <- "data/stars/data.csv"
69 D <- na.omit(read.csv(csvfile))
70 df <- D[(D$Multiple != '-') & (D$Multiple != 'W'),]
71 T <- deg.rad(90 - df$GLAT)
72 P <- deg.rad(df$GLON)
73 G <- factor(df$Multiple, levels=c("A", "D", "I", "R", "S"),
74           labels=c("AIR", "D", "AIR", "AIR", "S"))
75 data <- subset(df, select=-c(HR, GLAT, GLON, Multiple, SpType, UB))[,-(2:8)]
76 dim(data)
77 table(G)
78
79 plt <- plot.sphere(T, P, col=as.numeric(G)+1, cex=1.3)
80 plot.normal.shere(T, P, col=as.numeric(G)+1)
81
82 # -----
83 # Spherical Discriminant Analysis
84 # -----
85 # Generated data
86 gr <- rep(1:3, each=n)
87 X <- sphere.coord(theta, phi)
88 da <- vmfda.pred(X, X, gr)
89 metrics(da, gr)
90 confusion_matrix(da, gr)
91
92 w <- 12 * (da==gr) + 4
93 plt <- plot.sphere(theta, phi, cex=1.3, col=da+1, pch=w)
94 plot.normal.shere(theta, phi, col=da+1, pch=w)
95
96 # Star data
97 X <- sphere.coord(T, P)
98 da <- vmfda.pred(X, X, as.numeric(G))
99 metrics(da, as.numeric(G))
100 confusion_matrix(da, G)
101
102 w <- 12 * (da==as.numeric(G)) + 4
103 plt <- plot.sphere(T, P, cex=1.3, col=da+1, pch=w)
104 plot.normal.shere(T, P, col=da+1, pch=w)
105
106 # -----
107 # Spherical Logistic Regression
108 # -----

```

```

109 # Generated data
110 gr <- rep(1:3, each=n)
111 X <- as.data.frame(sphere.coord(theta, phi))
112 m <- nnet::multinom(gr ~ ., data=X)
113 summary(m)
114 pred <- predict(m)
115 metrics(pred, gr)
116 confusion_matrix(pred, gr)
117
118 w <- 12 * (pred==gr) + 4
119 plt <- plot.sphere(theta, phi, cex=1.3, col=as.numeric(pred)+1, pch=w)
120 plot.normal.shere(theta, phi, col=as.numeric(pred)+1, pch=w)
121
122 # Star data
123 X <- as.data.frame(sphere.coord(T, P))
124 m <- nnet::multinom(G ~ ., data=X)
125 summary(m)
126 pred <- predict(m)
127 metrics(pred, G)
128 confusion_matrix(pred, G)
129
130 w <- 12 * (pred==G) + 4
131 plt <- plot.sphere(T, P, cex=1.3, col=as.numeric(pred)+1, pch=w)
132 plot.normal.shere(T, P, col=as.numeric(pred)+1, pch=w)
133
134 # Star data with additional variables
135 x <- as.data.frame(sphere.coord(T, P))
136 X <- cbind(x, data)
137 m <- nnet::multinom(G ~ . - DEs - DEem - pmRA - RAm - RAh, data=X, maxit=200)
138 z <- summary(m)$coefficients / summary(m)$standard.errors
139 p <- 2 * pnorm(abs(z), 0, 1, lower.tail=FALSE)
140 signif(t(p), 4)
141 summary(m)
142 metrics(pred, G)
143 confusion_matrix(pred, G)
144
145 pred <- predict(m)
146 w <- 12 * (pred==G) + 4
147 plt <- plot.sphere(T, P, cex=1.3, col=as.numeric(pred)+1, pch=w)
148 plot.normal.shere(T, P, col=as.numeric(pred)+1, pch=w)
149
150 # -----
151 # Comparison with linear classification
152 # -----
153 x <- as.data.frame(norm.coord(T, P))
154 X <- cbind(x, data)
155 m <- nnet::multinom(G ~ . - DEs - DEem - pmRA - RAm - RAh, data=X)
156 summary(m)
157 pred <- predict(m)
158 metrics(pred, G)
159 confusion_matrix(pred, G)
160
161 w <- 12 * (pred==G) + 4
162 plt <- plot.sphere(T, P, cex=1.3, col=as.numeric(pred)+1, pch=w)
163 plot.normal.shere(T, P, col=as.numeric(pred)+1, pch=w)

```

```

1 # -----
2 # Generate cylindrical data

```

```

3  # -----
4  # generate data with 3 groups
5  set.seed(928)
6  n <- 100
7  theta <- c(rvonmises(n, circular(5.3), 5),
8             rvonmises(n, circular(3.9), 3),
9             rvonmises(n, circular(1.2), 4))
10 y <- c(rnorm(n, 3, 4), rnorm(n, 2.5, 2), rnorm(n, 7.5, 3))
11 gr <- rep(2:4, each=n)
12
13 plt <- plot.cylinder(theta, y, gr, cex=1.1)
14 plot.normal.cylinder(theta, y, gr)
15
16 x1 <- c(rnorm(n, 2.5, 1.4), rnorm(n, 7.6, 1.3), rnorm(n, 7.4, 1.4))
17 x2 <- c(rnorm(n, 4.8, 3), rnorm(n, 2.1, 1.6), rnorm(n, 7, 1.5))
18 plot(x1, x2, pch=16, cex=.8, col=gr)
19
20 # -----
21 # EM Clustering (Cylindrical)
22 # -----
23 # EM vMN
24 set.seed(4521)
25 res <- EMvMN(theta, y, 3)
26 pred <- res$groups
27 table(pred)
28 res$Iter
29 res$logL
30 res$pi.k
31 signif(res$param, 4)
32
33 plt <- plot.cylinder(theta, y, pred+1, cex=1.1)
34 M <- Reduce(rbind, by(cbind(theta, y), pred, cyl.mean))
35 plt$points3d(cylinder.coord(M[,1], M[,2]), pch='x', cex=2, col=2:4)
36
37 plot.normal.cylinder(theta, y, pred+1)
38 points(circ.break(M[,1]), M[,2], col=2:4, pch='x', cex=1.3)
39
40 # EM SWG
41 set.seed(4521)
42 res <- EM.SWG(theta, y, 3, max.iter=100)
43 pred <- res$groups
44 table(pred)
45 res$Iter
46 res$logL
47 res$pi.k
48 signif(sapply(res$param, '[' , 1), 4)
49
50 plt <- plot.cylinder(theta, y, pred+1, cex=1.1)
51 M <- Reduce(rbind, by(cbind(theta, y), pred, cyl.mean))
52 plt$points3d(cylinder.coord(M[,1], M[,2]), pch='x', cex=2, col=2:4)
53
54 plot.normal.cylinder(theta, y, pred+1)
55 points(circ.break(M[,1]), M[,2], col=2:4, pch='x', cex=1.3)
56
57 # -----
58 # K-means Clustering on Cylinder
59 # -----
60 # Fixed cyk-means
61 set.seed(2772)
62 res <- cyk.means(theta, y, 3, fixed=TRUE)

```

```

63 param <- cbind(N=table(res$pred), round(Reduce(rbind, res$centers), 3))
64 param[,2] <- circ.break(param[,2])
65 res$iter
66 param
67
68 plt <- plot.cylinder(theta, y, res$pred+1, cex=1.1)
69 plt$points3d(cylinder.coord(param[,2], param[,3]), pch='x', cex=2, col=2:4)
70
71 plot.normal.cylinder(theta, y, res$pred+1)
72 points(param[,2], param[,3], col=2:4, pch='x', cex=1.3)
73
74 # cyk-means
75 set.seed(2772)
76 res <- cyk.means(theta, y, 3, fixed=FALSE)
77 param <- cbind(N=table(res$pred), round(Reduce(rbind, res$centers), 3))
78 param[,2] <- circ.break(param[,2])
79 res$iter
80 param
81
82 plt <- plot.cylinder(theta, y, res$pred+1, cex=1.1)
83 plt$points3d(cylinder.coord(param[,2], param[,3]), pch='x', cex=2, col=2:4)
84
85 plot.normal.cylinder(theta, y, res$pred+1)
86 points(param[,2], param[,3], col=2:4, pch='x', cex=1.3)
87
88 # -----
89 # Linear Clustering (on cylinder)
90 # -----
91 set.seed(5264)
92 X <- cbind(theta, y)
93 res <- kmeans(X, 3, iter.max=50)
94 param <- cbind(N=table(res$cluster), round(res$centers, 3))
95 res$iter
96 param
97
98 plt <- plot.cylinder(theta, y, res$cluster+1, cex=1.1)
99 plt$points3d(cylinder.coord(param[,2], param[,3]), pch='x', cex=2, col=2:4)
100
101 plot.normal.cylinder(theta, y, res$cluster+1)
102 points(param[,2], param[,3], col=2:4, pch='x', cex=1.3)
103
104 # -----
105 # Real Cylinder Data
106 # -----
107 # all data
108 csvfile <- "data/meteo/data.csv"
109 df <- na.omit(read.csv(csvfile))
110 df$Date <- ymd_hm(df$Date)
111 T <- df$Wind.dir
112 G <- factor(month(df$Date), c(12, 1:11),
113             rep(c('Win', 'Spr', 'Sum', 'Aut'), each=3))
114 dim(df)
115 table(G)
116
117 pc <- prcomp(df[-c(1, 6)], scale=TRUE)
118 summary(pc)
119 signif(pc$rot[,1:2], 3)
120 Y <- pc$x[,1]
121
122 # mean daily data

```

```

123 csvfile <- "data/meteo/mean_data.csv"
124 dfm <- read.csv(csvfile)
125 dfm$Date <- ymd(dfm$Date)
126 Tm <- dfm$Wind.dir
127 Gm <- factor(month(dfm$Date), c(12, 1:11),
128             rep(c('Win', 'Spr', 'Sum', 'Aut'), each=3))
129 dim(dfm)
130 table(G)
131
132 pc <- prcomp(dfm[-c(1, 6)], scale=TRUE)
133 summary(pc)
134 signif(pc$rot[,1:2], 3)
135 Ym <- pc$x[,1]
136
137 # plots
138 c.plot(T, cex=.5, shrink=1.5, text.cex=1.2)
139 . <- sapply(df[-c(1, 6)], function(x)
140             plot.normal.cylinder(T, x, as.numeric(G)+1))
141 plot.normal.cylinder(T, Y, as.numeric(G)+1)
142
143 c.plot(Tm, cex=1.2, shrink=1.2, text.cex=1.2)
144 . <- sapply(dfm[-c(1, 6)], function(x)
145             plot.normal.cylinder(Tm, x, as.numeric(Gm)+1))
146 plot.normal.cylinder(Tm, Ym, as.numeric(Gm)+1)
147
148 . <- sapply(df[-c(1, 6)], function(x)
149             plot.cylinder(T, x, as.numeric(G)+1))
150 plt <- plot.cylinder(T, Y, as.numeric(G)+1)
151
152 . <- sapply(dfm[-c(1, 6)], function(x)
153             plot.cylinder(Tm, x, as.numeric(Gm)+1, cex=1.3))
154 plt <- plot.cylinder(Tm, Ym, as.numeric(Gm)+1, cex=1.3)
155
156 # -----
157 # Cylindrical Discriminant Analysis
158 # -----
159 # Cylindrical DA.vMN
160 # Generated data
161 pred <- DA.vMNCyl(theta, y, gr)
162 metrics(pred, gr)
163 confusion_matrix(pred, gr)
164
165 w <- 12 * (pred==gr) + 4
166 plt <- plot.cylinder(theta, y, as.numeric(pred)+1, cex=1.2, pch=w)
167 plot.normal.cylinder(theta, y, as.numeric(pred)+1, pch=w)
168
169 # Meteo data
170 pred <- DA.vMNCyl(T, Y, G)
171 metrics(pred, G)
172 confusion_matrix(pred, G)
173
174 w <- 12 * (pred==G) + 4
175 plt <- plot.cylinder(T, Y, as.numeric(pred)+1, pch=w)
176 plot.normal.cylinder(T, Y, as.numeric(pred)+1, pch=w)
177
178 # Daily Meteo data
179 pred <- DA.vMNCyl(Tm, Ym, Gm)
180 metrics(pred, Gm)
181 confusion_matrix(pred, Gm)
182

```

```

183 w <- 12 * (pred==Gm) + 4
184 plt <- plot.cylinder(Tm, Ym, as.numeric(pred)+1, cex=1.2, pch=w)
185 plot.normal.cylinder(Tm, Ym, as.numeric(pred)+1, pch=w)
186
187 # Cylindrical Mutli DA.vMN
188 # Meteo data
189 P <- sapply(df[-c(1, 6)], DA.vMNCyl, theta=T, group=G)
190 X <- as.data.frame(P)
191 m <- nnet::multinom(G ~ ., data=X, maxit=200)
192 z <- summary(m)$coefficients / summary(m)$standard.errors
193 p <- 2 * pnorm(abs(z), 0, 1, lower.tail=FALSE)
194 summary(m)
195 signif(t(p), 4)
196
197 pred <- predict(m)
198 metrics(pred, G)
199 confusion_matrix(pred, G)
200
201 w <- 12 * (pred==G) + 4
202 plt <- plot.cylinder(T, Y, as.numeric(pred)+1, pch=w)
203 plot.normal.cylinder(T, Y, as.numeric(pred)+1, pch=w)
204
205 # Daily Meteo data
206 P <- sapply(dfm[-c(1, 6)], DA.vMNCyl, theta=Tm, group=Gm)
207 X <- as.data.frame(P)
208 m <- nnet::multinom(Gm ~ ., data=X, maxit=200)
209 z <- summary(m)$coefficients / summary(m)$standard.errors
210 p <- 2 * pnorm(abs(z), 0, 1, lower.tail=FALSE)
211 summary(m)
212 signif(t(p), 4)
213
214 pred <- predict(m)
215 metrics(pred, Gm)
216 confusion_matrix(pred, Gm)
217
218 w <- 12 * (pred==Gm) + 4
219 plt <- plot.cylinder(Tm, Ym, as.numeric(pred)+1, cex=1.2, pch=w)
220 plot.normal.cylinder(Tm, Ym, as.numeric(pred)+1, pch=w)
221
222 # Cylindrical DA.SWG
223 # Generated data
224 pred <- DA.SWGcyl(theta, y, gr)
225 metrics(pred, gr)
226 confusion_matrix(pred, gr)
227
228 w <- 12 * (pred==gr) + 4
229 plt <- plot.cylinder(theta, y, as.numeric(pred)+1, cex=1.2, pch=w)
230 plot.normal.cylinder(theta, y, pred, pch=w)
231
232 # Meteo data
233 pred <- DA.SWGcyl(T, Y, G)
234 metrics(pred, G)
235 confusion_matrix(pred, G)
236
237 w <- 12 * (pred==G) + 4
238 plt <- plot.cylinder(T, Y, as.numeric(pred)+1, pch=w)
239 plot.normal.cylinder(T, Y, as.numeric(pred)+1, pch=w)
240
241 # Daily Meteo data
242 pred <- DA.SWGcyl(Tm, dfm[-c(1, 6)], Gm, tol=1e-5)

```

```

243 metrics(pred, Gm)
244 confusion_matrix(pred, Gm)
245
246 w <- 12 * (pred==Gm) + 4
247 plt <- plot.cylinder(Tm, Ym, as.numeric(pred)+1, cex=1.2, pch=w)
248 plot.normal.cylinder(Tm, Ym, as.numeric(pred)+1, pch=w)
249
250 # -----
251 # Cylindrical Logistic Regression
252 # -----
253 # Generated data
254 X <- as.data.frame(cylinder.coord(theta, y))
255 m <- nnet::multinom(gr ~ ., data=X)
256 summary(m)
257 pred <- predict(m)
258 metrics(pred, gr)
259 confusion_matrix(pred, gr)
260
261 w <- 12 * (pred==gr) + 4
262 plt <- plot.cylinder(theta, y, as.numeric(pred)+1, cex=1.2, pch=w)
263 plot.normal.cylinder(theta, y, pred, pch=w)
264
265 # Meteo data
266 X <- as.data.frame(cbind(cos(T), sin(T), df[-c(1, 6)]))
267 m <- nnet::multinom(G ~ . - Dew.temp, data=X, maxit=200)
268 z <- summary(m)$coefficients / summary(m)$standard.errors
269 p <- 2 * pnorm(abs(z), 0, 1, lower.tail=FALSE)
270 summary(m)
271 signif(t(p), 4)
272
273 pred <- predict(m)
274 metrics(pred, G)
275 confusion_matrix(pred, G)
276
277 w <- 12 * (pred==G) + 4
278 plt <- plot.cylinder(T, Y, as.numeric(pred)+1, pch=w)
279 plot.normal.cylinder(T, Y, as.numeric(pred)+1, pch=w)
280
281 # Daily Meteo data
282 X <- as.data.frame(cbind(cos(Tm), sin(Tm), dfm[-c(1, 6)]))
283 m <- nnet::multinom(Gm ~ . - Dew.temp, data=X, maxit=200)
284 z <- summary(m)$coefficients / summary(m)$standard.errors
285 p <- 2 * pnorm(abs(z), 0, 1, lower.tail=FALSE)
286 summary(m)
287 signif(t(p), 4)
288
289 pred <- predict(m)
290 metrics(pred, Gm)
291 confusion_matrix(pred, Gm)
292
293 w <- 12 * (pred==Gm) + 4
294 plt <- plot.cylinder(Tm, Ym, as.numeric(pred)+1, cex=1.2, pch=w)
295 plot.normal.cylinder(Tm, Ym, as.numeric(pred)+1, pch=w)
296
297 # -----
298 # Comparison with linear classification
299 # -----
300 # Meteo data
301 X <- df[-1]
302 m <- nnet::multinom(G ~ . - Dew.temp, data=X, maxit=200)

```

```

303 z <- summary(m)$coefficients / summary(m)$standard.errors
304 p <- 2 * pnorm(abs(z), 0, 1, lower.tail=FALSE)
305 summary(m)
306 signif(t(p), 4)
307
308 pred <- predict(m)
309 metrics(pred, G)
310 confusion_matrix(pred, G)
311
312 w <- 12 * (pred==G) + 4
313 plt <- plot.cylinder(T, Y, as.numeric(pred)+1, pch=w)
314 plot.normal.cylinder(T, Y, as.numeric(pred)+1, pch=w)
315
316 # Daily Meteo data
317 X <- dfm[-1]
318 m <- nnet::multinom(Gm ~ . - Dew.temp, data=X, maxit=200)
319 z <- summary(m)$coefficients / summary(m)$standard.errors
320 p <- 2 * pnorm(abs(z), 0, 1, lower.tail=FALSE)
321 summary(m)
322 signif(t(p), 4)
323
324 pred <- predict(m)
325 metrics(pred, Gm)
326 confusion_matrix(pred, Gm)
327
328 w <- 12 * (pred==Gm) + 4
329 plt <- plot.cylinder(Tm, Ym, as.numeric(pred)+1, cex=1.2, pch=w)
330 plot.normal.cylinder(Tm, Ym, as.numeric(pred)+1, pch=w)
331
332 # -----
333 # Decision Trees for directional data
334 # -----
335 # Simulated data
336 c.plot(theta, cex=1.5, shrink=1.2, text.cex=1.3)
337 param <- rbind(c(mu=5.3, kappa=5), c(mu=3.9, kappa=3), c(mu=1.2, kappa=4))
338 rownames(param) <- unique(gr)
339 . <- sapply(unique(gr), function(i)
340     tree.gr.plot(theta[gr==i], mu=param[i-1, 1], kappa=param[i-1, 2],
341     cex=1.7, shrink=1.55, text.cex=1.3))
342
343 # vMises model
344 D <- data.frame(theta, y, x1, x2, gr)
345 vm <- circrtree(theta ~ rep(1, length(theta)), data=D,
346     response_range=c(0, 2*pi))
347 plot(vm)
348
349 # vMises with tree regression
350 m <- circrtree(theta ~ x1 + x2, data=D, response_range=c(0, 2*pi))
351 m
352 plot(m)
353
354 pred <- cbind(theta, predict(m))
355 . <- lapply(split(pred, pred$mu), function(X)
356     tree.gr.plot(X$theta, mu=X$mu[1], kappa=X$kappa[1],
357     cex=1.7, shrink=1.55, text.cex=1.3))
358
359 # Meteorological data
360 m <- circrtree(Wind.dir ~ ., data=dfm[-1], response_range=c(0, 2*pi))
361 m
362 plot(m)

```

```

363
364 pred <- cbind(theta=dfm$Wind.dir, predict(m))
365 . <- lapply(split(pred, pred$mu), function(X)
366     tree.gr.plot(X$theta, mu=X$mu[1], kappa=X$kappa[1],
367     cex=1.7, shrink=1.55, text.cex=1.3))
368
369 # predict on all data
370 pred <- data.frame(cbind(theta=df$Wind.dir, predict(m, df[, -5])))
371 . <- lapply(split(pred, pred$mu), function(X)
372     tree.gr.plot(X$theta, mu=X$mu[1], kappa=X$kappa[1],
373     cex=.2, shrink=1.55, text.cex=1.3))

```

```

1  # -----
2  # Simulations
3  # -----
4
5  # -----
6  # Clustering on Sphere
7  # -----
8  orig.param <- cbind(rbind(c(1.5, 10), c(0, 3), c(1.8, 8)),
9                      rbind(c(4.5, 25), c(3, 5), c(5.2, 8)) )
10 colnames(orig.param) <- c('mu_theta', 'kappa_theta', 'mu_phi', 'kappa_phi')
11 m <- sphere.coord(orig.param[, 'mu_theta'], orig.param[, 'mu_phi'])
12 set.seed(5682)
13 M <- 1000
14
15 # EM Clustering
16 em.sphere.sim <- function(orig.param, n=100, ...){
17     theta <- c(rvonmises(n, circular(orig.param[1, 1]), orig.param[1, 2]),
18               rvonmises(n, circular(orig.param[2, 1]), orig.param[2, 2]),
19               rvonmises(n, circular(orig.param[3, 1]), orig.param[3, 2]))
20     phi <- c(rvonmises(n, circular(orig.param[1, 3]), orig.param[1, 4]),
21             rvonmises(n, circular(orig.param[2, 3]), orig.param[2, 4]),
22             rvonmises(n, circular(orig.param[3, 3]), orig.param[3, 4]))
23     res <- mixvmf.mle(sphere.coord(theta, phi), 3, n.start=10)
24     param <- cbind(N=table(res$pred), round(cbind(coord2ang(res$param[, 1:3]),
25                                                  res$param[, 4:5]), 3))
26     return(list(iter=res$iter, param=param))
27 }
28 em.sphere.res <- lapply(1:M, em.sphere.sim, orig.param=orig.param, n=100)
29 em.sphere.iter <- sapply(em.sphere.res, '[[', 1)
30 em.sphere.mu_theta <- t(sapply(em.sphere.res, function(x) x[[2]][, 'theta']))
31 em.sphere.mu_phi <- t(sapply(em.sphere.res, function(x) x[[2]][, 'phi']))
32 em.sphere.dist <- rowSums(sapply(1:3, function(i)
33     sphere.dist(em.sphere.mu_theta[, i], em.sphere.mu_phi[, i], m[i, ])))
34
35 # k-means Clustering
36 km.sphere.sim <- function(orig.param, n=100, ...){
37     theta <- c(rvonmises(n, circular(orig.param[1, 1]), orig.param[1, 2]),
38               rvonmises(n, circular(orig.param[2, 1]), orig.param[2, 2]),
39               rvonmises(n, circular(orig.param[3, 1]), orig.param[3, 2]))
40     phi <- c(rvonmises(n, circular(orig.param[1, 3]), orig.param[1, 4]),
41             rvonmises(n, circular(orig.param[2, 3]), orig.param[2, 4]),
42             rvonmises(n, circular(orig.param[3, 3]), orig.param[3, 4]))
43     res <- sphere.kmeans(theta, phi, 3)
44     X0 <- by(sphere.coord(theta, phi), res$pred, sphere.mean)
45     mu <- matrix(unlist(X0), ncol=4, byrow=TRUE)[-4, ]
46     param <- cbind(N=table(res$pred),

```

```

47     round(cbind(coord2ang(mu), res$param[,4:5]), 3))
48     return(list(iter=res$iter, param=param))
49 }
50 km.sphere.res <- lapply(1:M, km.sphere.sim, orig.param=orig.param, n=100)
51 km.sphere.iter <- sapply(km.sphere.res, '[', 1)
52 km.sphere.mu_theta <- t(sapply(km.sphere.res, function(x) x[[2]][, 'theta']))
53 km.sphere.mu_phi <- t(sapply(km.sphere.res, function(x) x[[2]][, 'phi']))
54 km.sphere.dist <- rowSums(sapply(1:3, function(i)
55     sphere.dist(km.sphere.mu_theta[,i], km.sphere.mu_phi[,i], m[i,])))
56
57 # Linear k-means Clustering
58 Lclust.sphere.sim <- function(orig.param, n=100, ...){
59     theta <- c(rvonmises(n, circular(orig.param[1, 1]), orig.param[1, 2]),
60         rvonmises(n, circular(orig.param[2, 1]), orig.param[2, 2]),
61         rvonmises(n, circular(orig.param[3, 1]), orig.param[3, 2]))
62     phi <- c(rvonmises(n, circular(orig.param[1, 3]), orig.param[1, 4]),
63         rvonmises(n, circular(orig.param[2, 3]), orig.param[2, 4]),
64         rvonmises(n, circular(orig.param[3, 3]), orig.param[3, 4]))
65     res <- kmeans(norm.coord(theta, phi), 3, iter.max=50)
66     param <- cbind(N=table(res$cluster), round(res$centers, 3))
67     return(list(iter=res$iter, param=param))
68 }
69 Lclust.sphere.res <- lapply(1:M, Lclust.sphere.sim,
70     orig.param=orig.param, n=100)
71 Lclust.sphere.iter <- sapply(Lclust.sphere.res, '[', 1)
72 Lclust.sphere.mu_theta <- t(sapply(Lclust.sphere.res, function(x)
73     x[[2]][, 'theta']))
74 Lclust.sphere.mu_phi <- t(sapply(Lclust.sphere.res, function(x)
75     x[[2]][, 'phi']))
76 Lclust.sphere.dist <- rowSums(sapply(1:3, function(i)
77     sphere.dist(Lclust.sphere.mu_theta[,i], Lclust.sphere.mu_phi[,i], m[i,])))
78
79 # Comparison
80 h <- hist(c(em.sphere.iter, km.sphere.iter, Lclust.sphere.iter), plot=FALSE)
81 hist(Lclust.sphere.iter, h$breaks, col=scales::alpha("#3399ff", .7),
82     xlab='Iteracijos', ylab='Dažnis', main='')
83 hist(em.sphere.iter, h$breaks, col=scales::alpha("#cc5a7a", .7),
84     xlab='Iteracijos', ylab='Dažnis', add=TRUE)
85 hist(km.sphere.iter, h$breaks, col=scales::alpha("#52527a", .7),
86     xlab='Iteracijos', ylab='Dažnis', add=TRUE)
87 legend('topright', c('EM', 'k-means', 'Klasikinis'),
88     fill=scales::alpha(c("#cc5a7a", "#52527a", "#3399ff"), .7))
89
90 h <- hist(c(em.sphere.dist, km.sphere.dist, Lclust.sphere.dist), plot=FALSE)
91 hist(km.sphere.dist, h$breaks, col=scales::alpha("#52527a", .9),
92     xlab='Atstumas', ylab='Dažnis', main='')
93 hist(em.sphere.dist, h$breaks, col=scales::alpha("#cc5a7a", .7),
94     xlab='Atstumas', ylab='Dažnis', add=TRUE)
95 hist(Lclust.sphere.dist, h$breaks, col=scales::alpha("#3399ff", .5),
96     xlab='Atstumas', ylab='Dažnis', add=TRUE)
97 legend('topleft', c('EM', 'k-means', 'Klasikinis'),
98     fill=scales::alpha(c("#cc5a7a", "#52527a", "#3399ff"), .7))
99
100 iter <- c(em=mean(em.sphere.iter), kmeans=mean(km.sphere.iter),
101     classical=mean(Lclust.sphere.iter))
102 dist <- c(em=mean(em.sphere.dist), kmeans=mean(km.sphere.dist),
103     classical=mean(Lclust.sphere.dist))
104 round(rbind(iter, dist), 3)
105
106 # -----

```

```

107 # Classification on Sphere
108 # -----
109 set.seed(5682)
110 M <- 1000
111
112 # Discriminant Analysis
113 da.sphere.sim <- function(orig.param, n=100, ...){
114   theta <- c(rvonmises(n, circular(orig.param[1, 1]), orig.param[1, 2]),
115             rvonmises(n, circular(orig.param[2, 1]), orig.param[2, 2]),
116             rvonmises(n, circular(orig.param[3, 1]), orig.param[3, 2]))
117   phi <- c(rvonmises(n, circular(orig.param[1, 3]), orig.param[1, 4]),
118           rvonmises(n, circular(orig.param[2, 3]), orig.param[2, 4]),
119           rvonmises(n, circular(orig.param[3, 3]), orig.param[3, 4]))
120   gr <- rep(1:3, each=n)
121   X <- sphere.coord(theta, phi)
122   pred <- vmfda.pred(X, X, gr)
123   met <- metrics(pred, gr)['avg', ]
124   return(met)
125 }
126 da.sphere.res <- t(sapply(1:M, da.sphere.sim, orig.param=orig.param, n=100))
127
128 # Logistic Regression
129 lr.sphere.sim <- function(orig.param, n=100, ...){
130   theta <- c(rvonmises(n, circular(orig.param[1, 1]), orig.param[1, 2]),
131             rvonmises(n, circular(orig.param[2, 1]), orig.param[2, 2]),
132             rvonmises(n, circular(orig.param[3, 1]), orig.param[3, 2]))
133   phi <- c(rvonmises(n, circular(orig.param[1, 3]), orig.param[1, 4]),
134           rvonmises(n, circular(orig.param[2, 3]), orig.param[2, 4]),
135           rvonmises(n, circular(orig.param[3, 3]), orig.param[3, 4]))
136   gr <- rep(1:3, each=n)
137   X <- as.data.frame(sphere.coord(theta, phi))
138   m <- nnet::multinom(gr ~ ., data=X, trace=FALSE)
139   pred <- predict(m)
140   met <- metrics(pred, gr)['avg', ]
141   return(met)
142 }
143 lr.sphere.res <- t(sapply(1:M, lr.sphere.sim, orig.param=orig.param, n=100))
144
145 # Classical Logistic Regression
146 Lclass.sphere.sim <- function(orig.param, n=100, ...){
147   theta <- c(rvonmises(n, circular(orig.param[1, 1]), orig.param[1, 2]),
148             rvonmises(n, circular(orig.param[2, 1]), orig.param[2, 2]),
149             rvonmises(n, circular(orig.param[3, 1]), orig.param[3, 2]))
150   phi <- c(rvonmises(n, circular(orig.param[1, 3]), orig.param[1, 4]),
151           rvonmises(n, circular(orig.param[2, 3]), orig.param[2, 4]),
152           rvonmises(n, circular(orig.param[3, 3]), orig.param[3, 4]))
153   gr <- rep(1:3, each=n)
154   X <- as.data.frame(norm.coord(theta, phi))
155   m <- nnet::multinom(gr ~ ., data=X, trace=FALSE)
156   pred <- predict(m)
157   met <- metrics(pred, gr)['avg', ]
158   return(met)
159 }
160 Lclass.sphere.res <- t(sapply(1:M, Lclass.sphere.sim,
161                             orig.param=orig.param, n=100))
162
163 # Comparison
164 h <- hist(c(da.sphere.res[,1], lr.sphere.res[,1], Lclass.sphere.res[,1]),
165          plot=FALSE)
166 hist(lr.sphere.res[,1], h$breaks, col=scales::alpha("#52527a", .7),

```

```

167     xlab='TiksiLumas', ylab='Dažnis', main='')
168 hist(da.sphere.res[,1], h$breaks, col=scales::alpha("#cc5a7a", .7),
169     xlab='TiksiLumas', ylab='Dažnis', add=TRUE)
170 hist(Lclass.sphere.res[,1], h$breaks, col=scales::alpha("#3399ff", .7),
171     xlab='TiksiLumas', ylab='Dažnis', add=TRUE)
172 legend('topleft', c('DA', 'LR', 'Klasikinis'),
173     fill=scales::alpha(c("#cc5a7a", "#52527a", "#3399ff"), .7))
174
175 h <- hist(c(da.sphere.res[,2], lr.sphere.res[,2], Lclass.sphere.res[,2]),
176     plot=FALSE)
177 hist(lr.sphere.res[,2], h$breaks, col=scales::alpha("#52527a", .7),
178     xlab='Specifiškumas', ylab='Dažnis', main='')
179 hist(da.sphere.res[,2], h$breaks, col=scales::alpha("#cc5a7a", .7),
180     xlab='Specifiškumas', ylab='Dažnis', add=TRUE)
181 hist(Lclass.sphere.res[,2], h$breaks, col=scales::alpha("#3399ff", .7),
182     xlab='Specifiškumas', ylab='Dažnis', add=TRUE)
183 legend('topleft', c('DA', 'LR', 'Klasikinis'),
184     fill=scales::alpha(c("#cc5a7a", "#52527a", "#3399ff"), .7))
185
186 h <- hist(c(da.sphere.res[,3], lr.sphere.res[,3], Lclass.sphere.res[,3]),
187     plot=FALSE)
188 hist(lr.sphere.res[,3], h$breaks, col=scales::alpha("#52527a", .7),
189     xlab='Jautrumas', ylab='Dažnis', main='')
190 hist(da.sphere.res[,3], h$breaks, col=scales::alpha("#cc5a7a", .7),
191     xlab='Jautrumas', ylab='Dažnis', add=TRUE)
192 hist(Lclass.sphere.res[,3], h$breaks, col=scales::alpha("#3399ff", .7),
193     xlab='Jautrumas', ylab='Dažnis', add=TRUE)
194 legend('topleft', c('DA', 'LR', 'Klasikinis'),
195     fill=scales::alpha(c("#cc5a7a", "#52527a", "#3399ff"), .7))
196
197 h <- hist(c(da.sphere.res[,4], lr.sphere.res[,4], Lclass.sphere.res[,4]),
198     plot=FALSE)
199 hist(lr.sphere.res[,4], h$breaks, col=scales::alpha("#52527a", .7),
200     xlab='AUC', ylab='Dažnis', main='')
201 hist(da.sphere.res[,4], h$breaks, col=scales::alpha("#cc5a7a", .7),
202     xlab='AUC', ylab='Dažnis', add=TRUE)
203 hist(Lclass.sphere.res[,4], h$breaks, col=scales::alpha("#3399ff", .7),
204     xlab='AUC', ylab='Dažnis', add=TRUE)
205 legend('topleft', c('DA', 'LR', 'Klasikinis'),
206     fill=scales::alpha(c("#cc5a7a", "#52527a", "#3399ff"), .7))
207
208 round(rbind(DA=colMeans(da.sphere.res),
209     LR=colMeans(lr.sphere.res),
210     Klasikinis=colMeans(Lclass.sphere.res)), 3)
211
212 # -----
213 # Clustering on Cylinder
214 # -----
215 orig.param <- cbind(rbind(c(5.3, 5), c(3.9, 3), c(1.2, 4)),
216     rbind(c(3, 4), c(2.5, 2), c(7.5, 3)))
217 colnames(orig.param) <- c('mu_theta', 'kappa', 'mu_y', 'sigma')
218 set.seed(5682)
219 M <- 100
220
221 # EM vMN
222 emvmn.cyl.sim <- function(orig.param, n=100, i){
223     theta <- c(rvonmises(n, circular(orig.param[1, 1]), orig.param[1, 2]),
224         rvonmises(n, circular(orig.param[2, 1]), orig.param[2, 2]),
225         rvonmises(n, circular(orig.param[3, 1]), orig.param[3, 2]))
226     y <- c(rnorm(n, orig.param[1, 3], orig.param[1, 4]),

```

```

227         rnorm(n, orig.param[2, 3], orig.param[2, 4]),
228         rnorm(n, orig.param[3, 3], orig.param[3, 4]))
229     res <- EMvMN(theta, y, 3, trace=FALSE)
230     pred <- res$groups
231     M <- Reduce(rbind, by(cbind(theta, y), pred, cyl.mean))
232     M[,1] <- circ.break(M[,1])
233     return(list(iter=res$Iter, param=M))
234 }
235 emvmn.cyl.res <- lapply(1:M, emvmn.cyl.sim, orig.param=orig.param, n=100)
236 emvmn.cyl.iter <- sapply(emvmn.cyl.res, '[[', 1)
237 emvmn.cyl.mu_theta <- t(sapply(emvmn.cyl.res, function(x) x[[2]][, 'theta']))
238 emvmn.cyl.mu_y <- t(sapply(emvmn.cyl.res, function(x) x[[2]][, 'y']))
239 emvmn.cyl.dist <- rowSums(sapply(1:3, function(i)
240     cyl.dist1(emvmn.cyl.mu_theta[,i], emvmn.cyl.mu_y[,i],
241     orig.param[i, 1], orig.param[i, 3])))
242
243 # EM SWG
244 emswg.cyl.sim <- function(orig.param, n=100, i){
245     theta <- c(rvonmises(n, circular(orig.param[1, 1]), orig.param[1, 2]),
246     rvonmises(n, circular(orig.param[2, 1]), orig.param[2, 2]),
247     rvonmises(n, circular(orig.param[3, 1]), orig.param[3, 2]))
248     y <- c(rnorm(n, orig.param[1, 3], orig.param[1, 4]),
249     rnorm(n, orig.param[2, 3], orig.param[2, 4]),
250     rnorm(n, orig.param[3, 3], orig.param[3, 4]))
251     res <- EM.SWG(theta, y, 3, trace=FALSE)
252     pred <- res$groups
253     M <- Reduce(rbind, by(cbind(theta, y), pred, cyl.mean))
254     M[,1] <- circ.break(M[,1])
255     return(list(iter=res$Iter, param=M))
256 }
257 emswg.cyl.res <- lapply(1:M, emswg.cyl.sim, orig.param=orig.param, n=100)
258 emswg.cyl.iter <- sapply(emswg.cyl.res, '[[', 1)
259 emswg.cyl.mu_theta <- t(sapply(emswg.cyl.res, function(x) x[[2]][, 'theta']))
260 emswg.cyl.mu_y <- t(sapply(emswg.cyl.res, function(x) x[[2]][, 'y']))
261 emswg.cyl.dist <- rowSums(sapply(1:3, function(i)
262     cyl.dist1(emswg.cyl.mu_theta[,i], emswg.cyl.mu_y[,i],
263     orig.param[i, 1], orig.param[i, 3])))
264
265 # Fixed cyk-means
266 fcykm.cyl.sim <- function(orig.param, n=100, ...){
267     theta <- c(rvonmises(n, circular(orig.param[1, 1]), orig.param[1, 2]),
268     rvonmises(n, circular(orig.param[2, 1]), orig.param[2, 2]),
269     rvonmises(n, circular(orig.param[3, 1]), orig.param[3, 2]))
270     y <- c(rnorm(n, orig.param[1, 3], orig.param[1, 4]),
271     rnorm(n, orig.param[2, 3], orig.param[2, 4]),
272     rnorm(n, orig.param[3, 3], orig.param[3, 4]))
273     res <- cyk.means(theta, y, 3, fixed=TRUE)
274     param <- cbind(N=table(res$pred), round(Reduce(rbind, res$centers), 3))
275     param[,2] <- circ.break(param[,2])
276     return(list(iter=res$iter, param=param))
277 }
278 fcykm.cyl.res <- lapply(1:M, fcykm.cyl.sim, orig.param=orig.param, n=100)
279 fcykm.cyl.iter <- sapply(fcykm.cyl.res, '[[', 1)
280 fcykm.cyl.mu_theta <- t(sapply(fcykm.cyl.res, function(x) x[[2]][, 'theta']))
281 fcykm.cyl.mu_y <- t(sapply(fcykm.cyl.res, function(x) x[[2]][, 'y']))
282 fcykm.cyl.dist <- rowSums(sapply(1:3, function(i)
283     cyl.dist1(fcykm.cyl.mu_theta[,i], fcykm.cyl.mu_y[,i],
284     orig.param[i, 1], orig.param[i, 3])))
285
286 # cyk-means

```

```

287 cykm.cyl.sim <- function(orig.param, n=100, ...){
288   theta <- c(rvonmises(n, circular(orig.param[1, 1]), orig.param[1, 2]),
289             rvonmises(n, circular(orig.param[2, 1]), orig.param[2, 2]),
290             rvonmises(n, circular(orig.param[3, 1]), orig.param[3, 2]))
291   y <- c(rnorm(n, orig.param[1, 3], orig.param[1, 4]),
292         rnorm(n, orig.param[2, 3], orig.param[2, 4]),
293         rnorm(n, orig.param[3, 3], orig.param[3, 4]))
294   res <- cyk.means(theta, y, 3, fixed=FALSE)
295   param <- cbind(N=table(res$pred), round(Reduce(rbind, res$centers), 3))
296   param[,2] <- circ.break(param[,2])
297   return(list(iter=res$iter, param=param))
298 }
299 cykm.cyl.res <- lapply(1:M, cykm.cyl.sim, orig.param=orig.param, n=100)
300 cykm.cyl.iter <- sapply(cykm.cyl.res, '[[', 1)
301 cykm.cyl.mu_theta <- t(sapply(cykm.cyl.res, function(x) x[[2]][, 'theta']))
302 cykm.cyl.mu_y <- t(sapply(cykm.cyl.res, function(x) x[[2]][, 'y']))
303 cykm.cyl.dist <- rowSums(sapply(1:3, function(i)
304   cyl.dist1(cykm.cyl.mu_theta[,i], cykm.cyl.mu_y[,i],
305             orig.param[i, 1], orig.param[i, 3])))
306
307 # Classical k-means
308 Lclust.cyl.sim <- function(orig.param, n=100, ...){
309   theta <- c(rvonmises(n, circular(orig.param[1, 1]), orig.param[1, 2]),
310             rvonmises(n, circular(orig.param[2, 1]), orig.param[2, 2]),
311             rvonmises(n, circular(orig.param[3, 1]), orig.param[3, 2]))
312   y <- c(rnorm(n, orig.param[1, 3], orig.param[1, 4]),
313         rnorm(n, orig.param[2, 3], orig.param[2, 4]),
314         rnorm(n, orig.param[3, 3], orig.param[3, 4]))
315   X <- cbind(theta, y)
316   res <- kmeans(X, 3, iter.max=50)
317   param <- cbind(N=table(res$cluster), round(res$centers, 3))
318   return(list(iter=res$iter, param=param))
319 }
320 Lclust.cyl.res <- lapply(1:M, Lclust.cyl.sim, orig.param=orig.param, n=100)
321 Lclust.cyl.iter <- sapply(Lclust.cyl.res, '[[', 1)
322 Lclust.cyl.mu_theta <- t(sapply(Lclust.cyl.res, function(x) x[[2]][, 'theta']))
323 Lclust.cyl.mu_y <- t(sapply(Lclust.cyl.res, function(x) x[[2]][, 'y']))
324 Lclust.cyl.dist <- rowSums(sapply(1:3, function(i)
325   cyl.dist1(Lclust.cyl.mu_theta[,i], Lclust.cyl.mu_y[,i],
326             orig.param[i, 1], orig.param[i, 3])))
327
328 # Comparison
329 h <- hist(c(emvmn.cyl.iter, emswg.cyl.iter, fcykm.cyl.iter,
330           cykm.cyl.iter, Lclust.cyl.iter), plot=FALSE)
331 hist(emvmn.cyl.iter, h$breaks, col=scales::alpha("#cc5a7a", .7),
332      xlab='Iteracijos', ylab='Dažnis', main='', ylim=c(0,100))
333 hist(emswg.cyl.iter, h$breaks, col=scales::alpha("#52527a", .7),
334      xlab='Iteracijos', ylab='Dažnis', add=TRUE)
335 hist(fcykm.cyl.iter, h$breaks, col=scales::alpha("#a8d194", .7),
336      xlab='Iteracijos', ylab='Dažnis', add=TRUE)
337 hist(cykm.cyl.iter, h$breaks, col=scales::alpha("#3399ff", .7),
338      xlab='Iteracijos', ylab='Dažnis', add=TRUE)
339 hist(Lclust.cyl.iter, h$breaks, col=scales::alpha("black", .5),
340      xlab='Iteracijos', ylab='Dažnis', add=TRUE)
341 legend('topright',
342       c('EM.vMN', 'EM.SWG', 'f-cyk-means', 'cyk-means', 'Klasikinis'),
343       fill=scales::alpha(c("#cc5a7a", "#52527a", "#a8d194",
344                           "#3399ff", "black"), .7))
345
346 h <- hist(c(emvmn.cyl.dist, emswg.cyl.dist, fcykm.cyl.dist,

```

```

347     cykm.cyl.dist, Lclust.cyl.dist), plot=FALSE)
348 hist(emvmn.cyl.dist, h$breaks, col=scales::alpha("#cc5a7a", .7),
349       xlab='Atstumas', ylab='Dažnis', main='', ylim=c(0,100))
350 hist(emswg.cyl.dist, h$breaks, col=scales::alpha("#52527a", .7),
351       xlab='Atstumas', ylab='Dažnis', add=TRUE)
352 hist(fcykm.cyl.dist, h$breaks, col=scales::alpha("#a8d194", .7),
353       xlab='Atstumas', ylab='Dažnis', add=TRUE)
354 hist(cykm.cyl.dist, h$breaks, col=scales::alpha("#3399ff", .7),
355       xlab='Atstumas', ylab='Dažnis', add=TRUE)
356 hist(Lclust.cyl.dist, h$breaks, col=scales::alpha("black", .5),
357       xlab='Atstumas', ylab='Dažnis', add=TRUE)
358 legend('topright',
359       c('EM.vMN', 'EM.SWG', 'f-cyk-means', 'cyk-means', 'Klasikinis'),
360       fill=scales::alpha(c("#cc5a7a", "#52527a", "#a8d194",
361                             "#3399ff", "black"), .7))
362
363 iter <- c(emvmn=mean(emvmn.cyl.iter), emswg=mean(emswg.cyl.iter),
364          fcykmeans=mean(fcykm.cyl.iter), cykmeans=mean(cykm.cyl.iter),
365          classical=mean(Lclust.cyl.iter))
366 dist <- c(emvmn=mean(emvmn.cyl.dist), emswg=mean(emswg.cyl.dist),
367          fcykmeans=mean(fcykm.cyl.dist), cykmeans=mean(cykm.cyl.dist),
368          classical=mean(Lclust.cyl.dist))
369 round(rbind(iter, dist), 3)
370
371
372 # -----
373 # Classification on Cylinder
374 # -----
375 set.seed(5682)
376 M <- 100
377
378 # Cylindrical DA.vMN
379 davmn.cyl.sim <- function(orig.param, n=100, ...){
380   theta <- c(rvnmises(n, circular(orig.param[1, 1]), orig.param[1, 2]),
381             rvnmises(n, circular(orig.param[2, 1]), orig.param[2, 2]),
382             rvnmises(n, circular(orig.param[3, 1]), orig.param[3, 2]))
383   y <- c(rnorm(n, orig.param[1, 3], orig.param[1, 4]),
384         rnorm(n, orig.param[2, 3], orig.param[2, 4]),
385         rnorm(n, orig.param[3, 3], orig.param[3, 4]))
386   gr <- rep(2:4, each=n)
387   pred <- DA.vMNCyl(theta, y, gr)
388   met <- metrics(pred, gr)['avg', ]
389   return(met)
390 }
391 davmn.cyl.res <- t(sapply(1:M, davmn.cyl.sim, orig.param=orig.param, n=100))
392
393 # Cylindrical DA.SWG
394 daswg.cyl.sim <- function(orig.param, n=100, ...){
395   theta <- c(rvnmises(n, circular(orig.param[1, 1]), orig.param[1, 2]),
396             rvnmises(n, circular(orig.param[2, 1]), orig.param[2, 2]),
397             rvnmises(n, circular(orig.param[3, 1]), orig.param[3, 2]))
398   y <- c(rnorm(n, orig.param[1, 3], orig.param[1, 4]),
399         rnorm(n, orig.param[2, 3], orig.param[2, 4]),
400         rnorm(n, orig.param[3, 3], orig.param[3, 4]))
401   gr <- rep(2:4, each=n)
402   pred <- DA.SWGCyl(theta, y, gr, trace=FALSE)
403   met <- metrics(pred, gr)['avg', ]
404   return(met)
405 }
406 daswg.cyl.res <- t(sapply(1:M, daswg.cyl.sim, orig.param=orig.param, n=100))

```

```

407
408 # Cylindrical Logistic Regression
409 lr.cyl.sim <- function(orig.param, n=100, ...){
410   theta <- c(rvnmises(n, circular(orig.param[1, 1]), orig.param[1, 2]),
411             rvnmises(n, circular(orig.param[2, 1]), orig.param[2, 2]),
412             rvnmises(n, circular(orig.param[3, 1]), orig.param[3, 2]))
413   y <- c(rnorm(n, orig.param[1, 3], orig.param[1, 4]),
414         rnorm(n, orig.param[2, 3], orig.param[2, 4]),
415         rnorm(n, orig.param[3, 3], orig.param[3, 4]))
416   gr <- rep(2:4, each=n)
417   X <- as.data.frame(cylinder.coord(theta, y))
418   m <- nnet::multinom(gr ~ ., data=X, trace=FALSE)
419   pred <- predict(m)
420   met <- metrics(pred, gr)['avg', ]
421   return(met)
422 }
423 lr.cyl.res <- t(sapply(1:M, lr.cyl.sim, orig.param=orig.param, n=100))
424
425 # Classic Logistic Regression
426 Lclass.cyl.sim <- function(orig.param, n=100, ...){
427   theta <- c(rvnmises(n, circular(orig.param[1, 1]), orig.param[1, 2]),
428             rvnmises(n, circular(orig.param[2, 1]), orig.param[2, 2]),
429             rvnmises(n, circular(orig.param[3, 1]), orig.param[3, 2]))
430   y <- c(rnorm(n, orig.param[1, 3], orig.param[1, 4]),
431         rnorm(n, orig.param[2, 3], orig.param[2, 4]),
432         rnorm(n, orig.param[3, 3], orig.param[3, 4]))
433   gr <- rep(2:4, each=n)
434   X <- data.frame(theta, y)
435   m <- nnet::multinom(gr ~ ., data=X, maxit=200, trace=FALSE)
436   pred <- predict(m)
437   met <- metrics(pred, gr)['avg', ]
438   return(met)
439 }
440 Lclass.cyl.res <- t(sapply(1:M, Lclass.cyl.sim, orig.param=orig.param, n=100))
441
442 # Comparison
443 h <- hist(c(davmn.cyl.res[,1], daswg.cyl.res[,1], lr.cyl.res[,1],
444           Lclass.cyl.res[,1]), plot=FALSE)
445 hist(Lclass.cyl.res[,1], h$breaks, col=scales::alpha("#3399ff", .9),
446      xlab='Tiksiľumas', ylab='Dažnis', main='')
447 hist(davmn.cyl.res[,1], h$breaks, col=scales::alpha("#cc5a7a", .7),
448      xlab='Tiksiľumas', ylab='Dažnis', add=TRUE)
449 hist(daswg.cyl.res[,1], h$breaks, col=scales::alpha("#a8d194", .5),
450      xlab='Tiksiľumas', ylab='Dažnis', add=TRUE)
451 hist(lr.cyl.res[,1], h$breaks, col=scales::alpha("#52527a", .5),
452      xlab='Tiksiľumas', ylab='Dažnis', add=TRUE)
453 legend('topleft', c('DA.vMN', 'DA.SWG', 'LR', 'Klasikinis'),
454       fill=scales::alpha(c("#cc5a7a", "#a8d194", "#52527a", "#3399ff"), .7))
455
456 h <- hist(c(davmn.cyl.res[,2], daswg.cyl.res[,2], lr.cyl.res[,2],
457           Lclass.cyl.res[,2]), plot=FALSE)
458 hist(davmn.cyl.res[,2], h$breaks, col=scales::alpha("#cc5a7a", .9),
459      xlab='Specifiškus', ylab='Dažnis', main='')
460 hist(daswg.cyl.res[,2], h$breaks, col=scales::alpha("#a8d194", .7),
461      xlab='Specifiškus', ylab='Dažnis', add=TRUE)
462 hist(lr.cyl.res[,2], h$breaks, col=scales::alpha("#52527a", .5),
463      xlab='Specifiškus', ylab='Dažnis', add=TRUE)
464 hist(Lclass.cyl.res[,2], h$breaks, col=scales::alpha("#3399ff", .5),
465      xlab='Specifiškus', ylab='Dažnis', add=TRUE)
466 legend('topleft', c('DA.vMN', 'DA.SWG', 'LR', 'Klasikinis'),

```

```

467     fill=scales::alpha(c("#cc5a7a", "#a8d194", "#52527a", "#3399ff"), .7))
468
469 h <- hist(c(davmn.cyl.res[,3], daswg.cyl.res[,3], lr.cyl.res[,3],
470           Lclass.cyl.res[,3]), plot=FALSE)
471 hist(Lclass.cyl.res[,3], h$breaks, col=scales::alpha("#3399ff", .9),
472       xlab='Jautrumas', ylab='Dažnis', main='')
473 hist(davmn.cyl.res[,3], h$breaks, col=scales::alpha("#cc5a7a", .7),
474       xlab='Jautrumas', ylab='Dažnis', add=TRUE)
475 hist(daswg.cyl.res[,3], h$breaks, col=scales::alpha("#a8d194", .5),
476       xlab='Jautrumas', ylab='Dažnis', add=TRUE)
477 hist(lr.cyl.res[,3], h$breaks, col=scales::alpha("#52527a", .5),
478       xlab='Jautrumas', ylab='Dažnis', add=TRUE)
479 legend('topleft', c('DA.vMN', 'DA.SWG', 'LR', 'Klasikinis'),
480       fill=scales::alpha(c("#cc5a7a", "#a8d194", "#52527a", "#3399ff"), .7))
481
482 h <- hist(c(davmn.cyl.res[,4], daswg.cyl.res[,4], lr.cyl.res[,4],
483           Lclass.cyl.res[,4]), plot=FALSE)
484 hist(Lclass.cyl.res[,4], h$breaks, col=scales::alpha("#3399ff", .9),
485       xlab='AUC', ylab='Dažnis', main='')
486 hist(davmn.cyl.res[,4], h$breaks, col=scales::alpha("#cc5a7a", .7),
487       xlab='AUC', ylab='Dažnis', add=TRUE)
488 hist(daswg.cyl.res[,4], h$breaks, col=scales::alpha("#a8d194", .5),
489       xlab='AUC', ylab='Dažnis', add=TRUE)
490 hist(lr.cyl.res[,4], h$breaks, col=scales::alpha("#52527a", .5),
491       xlab='AUC', ylab='Dažnis', add=TRUE)
492 legend('topleft', c('DA.vMN', 'DA.SWG', 'LR', 'Klasikinis'),
493       fill=scales::alpha(c("#cc5a7a", "#a8d194", "#52527a", "#3399ff"), .7))
494
495 round(rbind(DA.vMN=colMeans(davmn.cyl.res),
496             DA.SWG=colMeans(daswg.cyl.res),
497             LR=colMeans(lr.cyl.res),
498             Klasikinis=colMeans(Lclass.cyl.res)), 3)

```