Vignette 02:

Example creation of *ellipsoid fencing* on sampled data

Ty Stanford, Maddison Mellow, et al.

Table of contents

1 Example data: Faircough et al. (2017) 2

2 Other resources and work-throughs 2

3 Data loading and wrangling 3

3.1 Set up R Session 3

3.2 Time use variables and isometric log-ratio (*ilr*) creation 5

4 Ellipsoid fencing 7

4.1 Estimated mean vector and var-covariance matrix 8

5 Inside or outside ellipsoid fencing calculations 9

5.1 Estimated contours on sampled data 11

5.1.1 Fencing for sample ilr data 11

5.1.2 Fencing for sample compositional data 13

5.2 Examine normality assumption: quantile plot 14

1 Example data: Faircough et al. (2017)

We will use the freely available Fairclough (2017) data[1] that is available in the `codaredistlm` R package.

The data at a glance:

- Examining children's daily movement behaviours and adiposity measures
- Year 5 children from a low-income community in northwest England
- Children's daily movement behaviours captured are: sleep, sedentary behaviour (sed), light/moderate/vigorous physical activity (which we combine for clarity of example)
- $n = 169$

2 Other resources and work-throughs

Also please note other more detailed and higher dimensional work-through are available in the supplementary material of Nikfarjam et al. (2024) data[2]

- Direct link to downloadable supp material `ppap447_Nikfarjam_suppl.zip`

[1] Fairclough et al. Fitness, fatness and the reallocation of time between children's daily movement behaviours: an analysis of compositional data. International Journal of Behavioral Nutrition and Physical Activity, 2017. 14(1): 64.

[2] Adel Nikfarjam et al. (2024). Quality Diversity Approaches for Time-Use Optimisation to Improve Health Outcomes. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '24). Association for Computing Machinery, New York, NY, USA, 1318–1326. https://doi.org/10.1145/3638529.3654085

3 Data loading and wrangling

3.1 Set up R Session

```r
# ---- libs -----

suppressPackageStartupMessages({
  library("simplexity") # see https://github.com/tystan/simplexity
  library("compositions")
  library("codaredistlm")

  library("knitr")
  library("dplyr")
  library("tidyr")
})


# ---- constants -----

timeuse_labs <- c("sleep", "sed", "pa")
D <- length(timeuse_labs)
(ilr_labs <- paste0("ilr", 1:(D - 1)))
```

```
#> [1] "ilr1" "ilr2"
```

```r
# ---- load_data ----

# load some real data
data("fairclough", package = "codaredistlm")

fairclough_example <-
  fairclough %>%
  as_tibble(.) %>%
  mutate(pa = lpa + mpa + vpa) %>%
  select(sleep, sed, pa)


kable(head(fairclough_example))
```

sleep	sed	pa
483.2	548.4	408.4
531.7	508.4	399.9

sleep	sed	pa
567.5	456.7	415.8
573.0	526.1	340.9
564.0	481.4	394.5
546.7	525.1	368.2

3.2 Time use variables and isometric log-ratio (*ilr*) creation

We have $n = 169$ observed time-use compositions in the example data, x_i for $i = 1, 2, \ldots, n$, where:

$$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \end{bmatrix} = \begin{bmatrix} x_{i,sleep} \\ x_{i,sed} \\ x_{i,pa} \end{bmatrix}.$$

We will assume[3] that when we log-ratio (*ilr*) transform the compositional data these observations are taken from the *Gaussian distribution*. This means the corresponding *ilr*s,

$$z_i = \begin{bmatrix} z_{i1} \\ z_{i2} \end{bmatrix},$$

are 2-dimensional/variate Gaussian distributed, written:

$$Z_i \sim \mathcal{N}_2(\mu_z, \Sigma_z).$$

Now let's create the ilr transformation (with our orthogonal basis matrix of choice)

```r
# ---- create_ilrs -----



# this creates a normalised one-vs-rest squential binary
# partition for ilr() transformation
(sbp3_b0 <-
  matrix(
    c(
      +1,  0,
      -1, +1,
      -1, -1
    ),
    byrow = TRUE,
    ncol = 2,
    dimnames = list(timeuse_labs, ilr_labs)
  ))
```

```
#>        ilr1 ilr2
#> sleep     1    0
#> sed      -1    1
#> pa       -1   -1
```

[3] we will check this later

```r
(psi3_b0 <- compositions::gsi.buildilrBase(sbp3_b0))
```

```
#>             ilr1       ilr2
#> sleep  0.8164966  0.0000000
#> sed   -0.4082483  0.7071068
#> pa    -0.4082483 -0.7071068
```

```r
ilr_dat <- ilr(fairclough_example, V = psi3_b0)
head(ilr_dat)
```

```
#>            ilr1       ilr2
#> [1,] 0.01698691 0.20842528
#> [2,] 0.13458958 0.16974383
#> [3,] 0.21565732 0.06634242
#> [4,] 0.24686404 0.30681514
#> [5,] 0.21057073 0.14077038
#> [6,] 0.17782700 0.25099636
#> attr(,"class")
#> [1] "rmult"
```

```r
ilr_dat <- as.matrix(as.data.frame(ilr_dat))
str(ilr_dat)
```

```
#>  num [1:169, 1:2] 0.017 0.135 0.216 0.247 0.211 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : NULL
#>   ..$ : chr [1:2] "ilr1" "ilr2"
```

4 Ellipsoid fencing

Any observation $z \in R^{D-1}$ can be evaluated to determine whether it is within the $100{\times}p^{th}$ percentile (i.e., a probability/proportion) contour of a multivariate normal distribution through the following inequality:

$$\left(z-\mu_z\right)^T \Sigma_z^{-1}\left(z-\mu_z\right) \le \chi^2_{D-1}(p)$$

where $\chi^2_{D-1}(p)$ is the $100{\times}p^{th}$ percentile of the Chi squared distribution with $D-1$ degrees of freedom. Note $D=3$ and therefore $D-1=2$ in this case.

Let's use $p = 0.8$ - that is - we expect roughly 80% of sampled values to lie within the associated contour and 20% outside of it. Therefore the RHS of the above inequality can be written:

$$\left(z-\mu_z\right)^T \Sigma_z^{-1}\left(z-\mu_z\right) \le 3.218876$$

as $\chi^2_2(0.8) = 3.2188758$.

As we do not know the true values of μ_z and Σ_z we can estimate them as $\hat{\mu}_z$ and $\hat{\Sigma}_z$, respectively, from the $n = 169$ sampled ilr values.

4.1 Estimated mean vector and var-covariance matrix

Here are the estimated quantities below:

```r
(m_ilr <- colMeans(ilr_dat))
```

```
#>      ilr1      ilr2
#> 0.1747038 0.2002131
```

```r
(v_ilr <- var(ilr_dat))
```

```
#>             ilr1        ilr2
#> ilr1 0.007629762 0.002621386
#> ilr2 0.002621386 0.051246892
```

That is, written out:

$$\hat{\mu}_z = \begin{bmatrix} 0.1747 \\ 0.2002 \end{bmatrix},$$

$$\hat{\Sigma}_z = \begin{bmatrix} 0.0076 & 0.0026 \\ 0.0026 & 0.0512 \end{bmatrix},$$

and therefore using the below R output,

```r
solve(v_ilr)
```

```
#>            ilr1      ilr2
#> ilr1 133.410314 -6.824217
#> ilr2  -6.824217 19.862452
```

we have

$$\hat{\Sigma}_z^{-1} = \begin{bmatrix} 133.41 & -6.82 \\ -6.82 & 19.86 \end{bmatrix},$$

Therefore the inequality below only has one unknown in z which can be input to see if it *satisfies the inequality* or *not*.

$$\left(z - \hat{\mu}_z\right)^T \hat{\Sigma}_z^{-1} \left(z - \hat{\mu}_z\right) \leq 3.218876$$

5 Inside or outside ellipsoid fencing calculations

Using the `get_inequality_value()` function defined below, we can either calculate the LHS of the inequality (to compare to the threshold value) or directly calculate the percentile contour that point lies on (usage: `get_inequality_value(..., as_percentile = TRUE)`).

```r
# see: ?stats::mahalanobis
get_inequality_value <- function(dat, mean_vec, covar_mat, as_percentile = FALSE) {
  mdist <- mahalanobis(x = dat, center = mean_vec, cov = covar_mat)
  if (as_percentile) {
    return(100 * pchisq(mdist, df = ncol(covar_mat)))
  } else {
    return(mdist)
  }
}


ilr_df <-
  as_tibble(fairclough_example) %>%
  bind_cols(., ilr_dat) %>%
  mutate(
    lhs_ineq = get_inequality_value(
      ilr_dat,
      mean_vec = m_ilr,
      covar_mat = v_ilr,
      as_percentile = FALSE
    ),
    perc_from_mean = get_inequality_value(
      ilr_dat,
      mean_vec = m_ilr,
      covar_mat = v_ilr,
      as_percentile = TRUE
    )
  )

# print out first 10 rows
kable(ilr_df[1:10,], digits = c(1, 1, 1, 3, 3, 1, 1))
```

sleep	sed	pa	ilr1	ilr2	lhs_ineq	perc_from_mean
483.2	548.4	408.4	0.017	0.208	3.3	81.2
531.7	508.4	399.9	0.135	0.170	0.2	10.3
567.5	456.7	415.8	0.216	0.066	0.7	27.9
573.0	526.1	340.9	0.247	0.307	0.8	33.5

sleep	sed	pa	ilr1	ilr2	lhs_ineq	perc_from_mean
564.0	481.4	394.5	0.211	0.141	0.3	12.7
546.7	525.1	368.2	0.178	0.251	0.1	2.5
580.5	481.2	378.3	0.251	0.170	0.8	34.1
516.9	687.4	235.7	0.204	0.757	6.0	95.1
541.0	444.3	454.7	0.151	-0.016	0.9	37.4
549.7	620.0	270.3	0.241	0.587	3.2	79.9

5.1 Estimated contours on sampled data

Let's examine the estimated 80% fencing boundary.

5.1.1 Fencing for sample ilr data

Calculate point's inclusion or exclusion of the fence boundary:

```r
fence_p <- 80
ilr_df$ellipsoid_80_fence <-
  if_else(
    ilr_df$perc_from_mean < fence_p,
    "inside fence",
    "outside fence"
  )
ilr_df$ellipsoid_80_fence <-
  factor(ilr_df$ellipsoid_80_fence)

col_pal <- add_alpha(c("dodgerblue", "orange"), 0.3)
col_vec <-
  if_else(
    ilr_df$ellipsoid_80_fence == "inside fence",
    col_pal[1],
    col_pal[2]
  )


head(ilr_df)
```

```
#> # A tibble: 6 x 8
#>   sleep   sed    pa   ilr1   ilr2 lhs_ineq perc_from_mean ellipsoid_80_fence
#>   <dbl> <dbl> <dbl>  <dbl>  <dbl>    <dbl>          <dbl> <fct>
#> 1  483.  548.  408. 0.0170 0.208    3.34           81.2  outside fence
#> 2  532.  508.  400. 0.135  0.170    0.216          10.3  inside fence
#> 3  568.  457.  416. 0.216  0.0663   0.655          27.9  inside fence
#> 4  573   526.  341. 0.247  0.307    0.815          33.5  inside fence
#> 5  564   481.  394. 0.211  0.141    0.271          12.7  inside fence
#> 6  547.  525.  368. 0.178  0.251    0.0504          2.49 inside fence
```

Figure 1 below shows the sampled points (ilr transformed) inside and outside the 80% ellipsoid fencing. We can see approximately 80% of points are within the estimated 80% contour of the Gaussian distribution on the simplex.

```r
plot(ilr_df[, ilr_labs], col = col_vec, pch = 16, bty = "n")

# add the mean ilr as black plus sign
points(m_ilr[1], m_ilr[2], pch = "+", cex = 3)
```

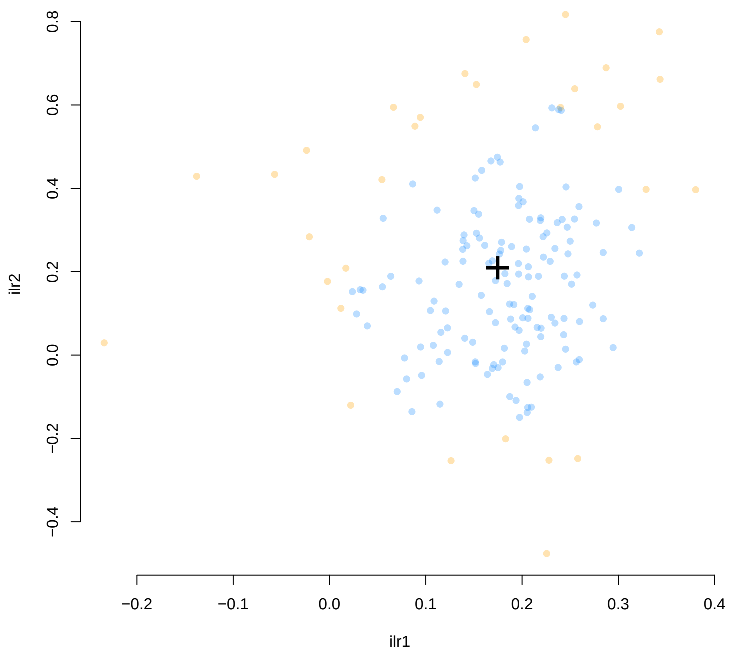

Figure 1: $n = 169$ compositions transformed to ilrs (z_1, z_2). Points are coloured blue if within the estimated 80% contour and orange if outside it.

5.1.2 Fencing for sample compositional data

From Figure 2, we see the compositions back-transformed to the simplex with their fencing categorisation.

```r
plot(acomp(ilr_df[, timeuse_labs]), col = col_vec,pch = 16)
```

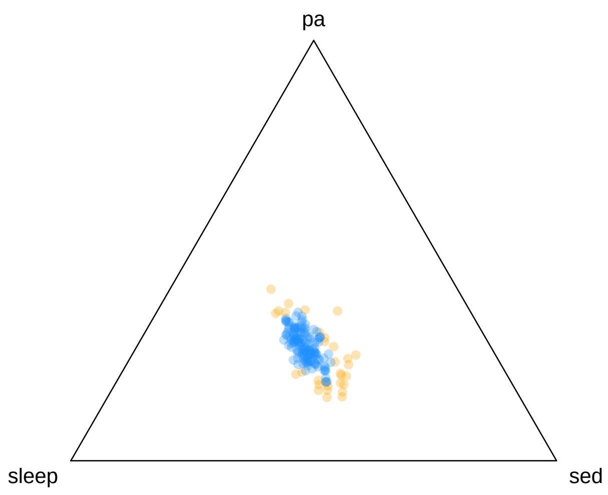

Figure 2: $n = 169$ points of $(x1, x2, x3)$=(sleep,sed,pa) sampled compositions. Points are coloured blue if within the estimated 80% contour and orange if outside it.

5.2 Examine normality assumption: quantile plot

We need to check that the previously estimated contours are appropriate for the sampled data. We can do this empirically.

Figure 3 shows the actual percentiles observed in the data, under the assumption of being from a

$$Z \sim \mathcal{N}_2(\hat{\mu}_z, \hat{\Sigma}_z).$$

distribution where $\hat{\mu}_z$ and $\hat{\Sigma}_z$ are given above. Figure 3 shows that there is a slightly larger density of observed points in each contour than what is theoretically expected. However this difference is slight and the assumption of multivariate normality is reasonable.

```
percentiles <- seq(5, 95, by = 5)
(obs_perc <- sapply(percentiles, function(x) 100 * mean(ilr_df$perc_from_mean <= x)))
```

```
#>  [1]  7.100592 12.426036 18.934911 24.260355 29.585799 36.094675 42.603550 48.520710
#>  [9] 52.662722 56.804734 60.946746 64.497041 68.047337 71.005917 76.923077 81.656805
#> [17] 85.798817 90.532544 94.674556
```

```
plot(
  0:100, 0:100,
  type = "n", bty = "n",
  xlab = "Theoretical CDF percentiles",
  ylab = "Actual CDF percentiles"
)
points(percentiles, obs_perc, type = "p", pch = 16, col = "magenta")
abline(a = 0, b = 1)
```

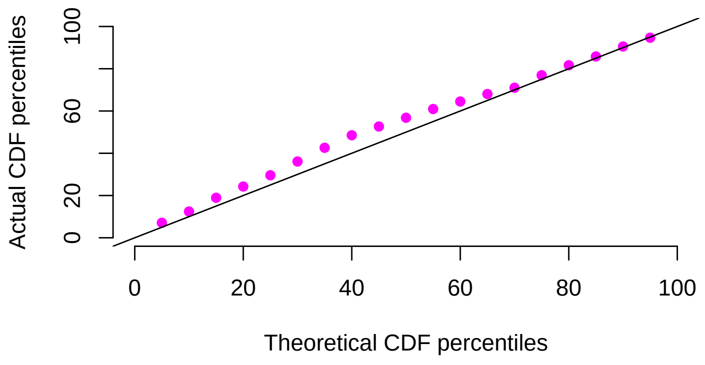

Figure 3: Proportion of sample points within the pth quantile contour $(f(z) \leq \chi^2_2(p))$ for the estimated Gaussian distribution for $p = 0.05, 0.10, ..., 0.95$