Visualization

Max Turgeon

STAT 4690–Applied Multivariate Analysis

Tidyverse

- For graphics, I personally prefer using `ggplot2` than base `R` functions.
  - Of course, you're free to use whatever you prefer!
- Therefore, I often use the `tidyverse` packages to prepare data for visualization
- Great resources:
  - The book *R for Data Science*
  - RStudio's cheatsheets

Pipe operator

- One of the important features of the `tidyverse` is the pipe operator %>%
- It takes the output of a function (or of an expression) and uses it as input for the next function (or expression)

```r
library(tidyverse)

count(mtcars, cyl)
# Or with the pipe
mtcars %>% count(cyl)
```

Pipe operator

- Note that the LHS (`mtcars`) becomes the first argument of the function appearing on the RHS (`count`)
- In more complex examples, where multiple transformations are applied one after another, the pipe operator improves readability and avoids creating too many intermediate variables.

Main tidyverse functions

- `mutate`: Create a new variable as a function of the other variables

```
mutate(mtcars, liters_per_100km = mpg/235.215)
```

- `filter`: Keep only rows for which some condition is `TRUE`

```
filter(mtcars, cyl %in% c(6, 8))
```

- `summarise`: Apply summary function to some variables. Often used with `group_by`.

```
mtcars %>% group_by(cyl) %>%
  summarise(avg_mpg = mean(mpg))
```

Data Visualization

Main principles

Why would we want to visualize data?

- Quality control
- Identify outliers
- Find patterns of interest (EDA)

Visualizing multivariate data

- To start, you can visualize multivariate data one variable at a time.
- Therefore, you can use the same visualizing tools you're likely familiar with.

Histogram i

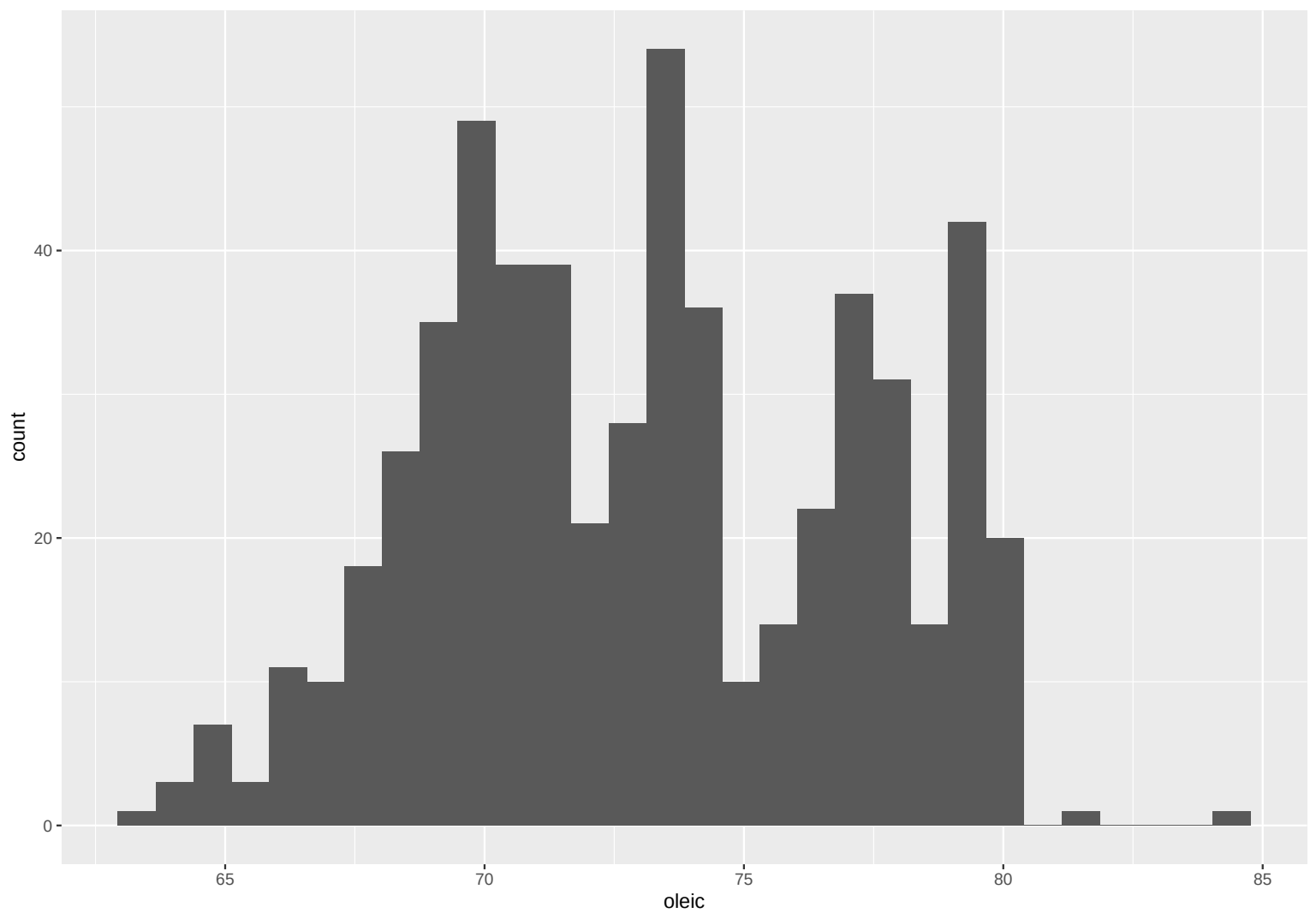
```
library(tidyverse)
library(dslabs)

dim(olive)
```

```
## [1] 572  10
```

```
olive %>%
  ggplot(aes(oleic)) +
  geom_histogram()
```

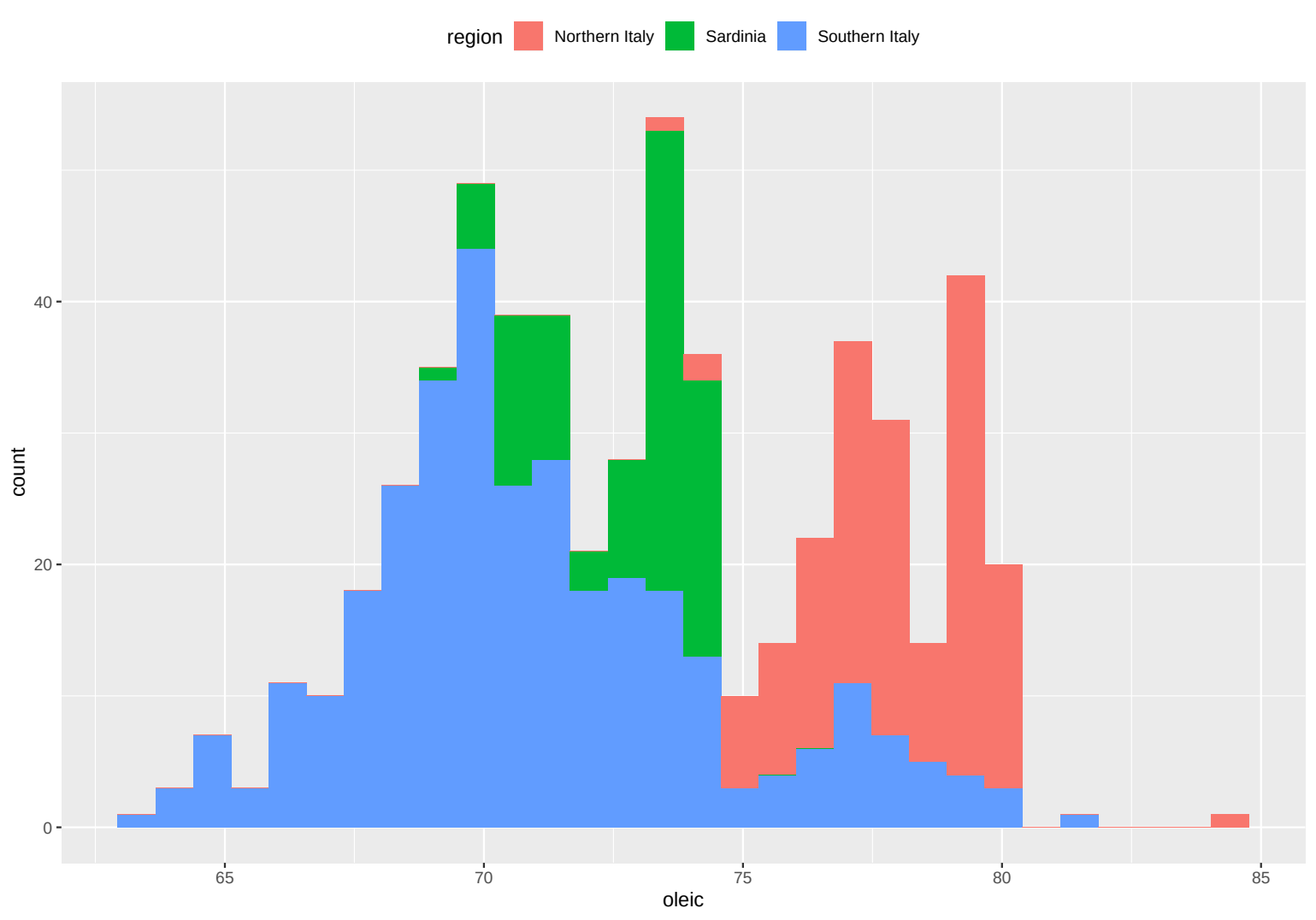
Histogram ii



Histogram iii

```
olive %>%
  ggplot(aes(oleic, fill = region)) +
  geom_histogram() +
  theme(legend.position = 'top')
```

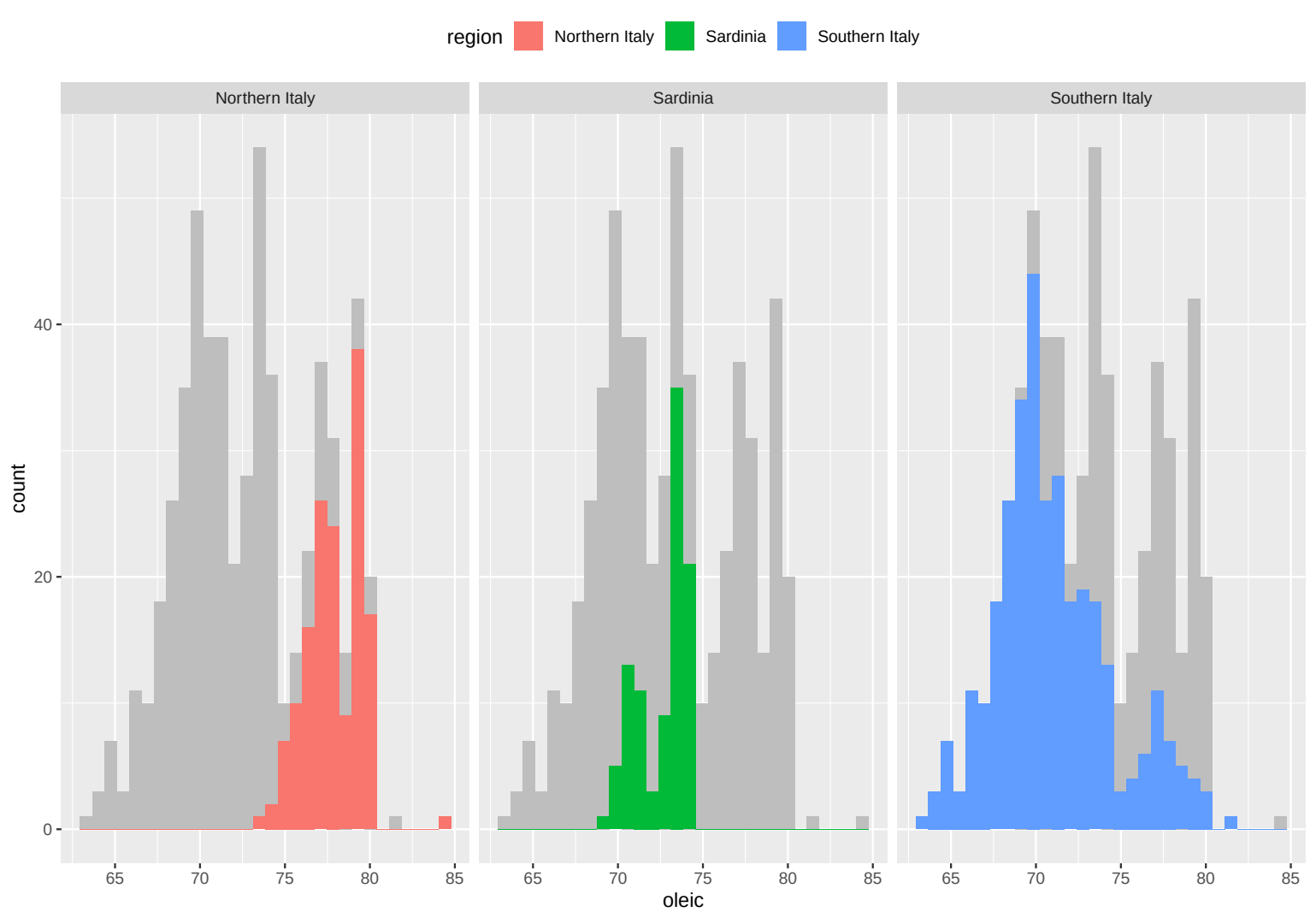
Histogram iv



Histogram v

```r
# Or with facets
olive_bg <- olive %>% dplyr::select(-region)
olive %>%
  ggplot(aes(oleic, fill = region)) +
  geom_histogram(data = olive_bg,
                 fill = 'grey') +
  geom_histogram() +
  facet_grid(. ~ region) +
  theme(legend.position = 'top')
```

Histogram vi



Density plot i

- Another way to estimate the density is with *kernel density estimators*.
- Let $X_1, \ldots, X_n$ be our IID sample. For K a non-negative function and $h > 0$ a smoothing parameter, we have

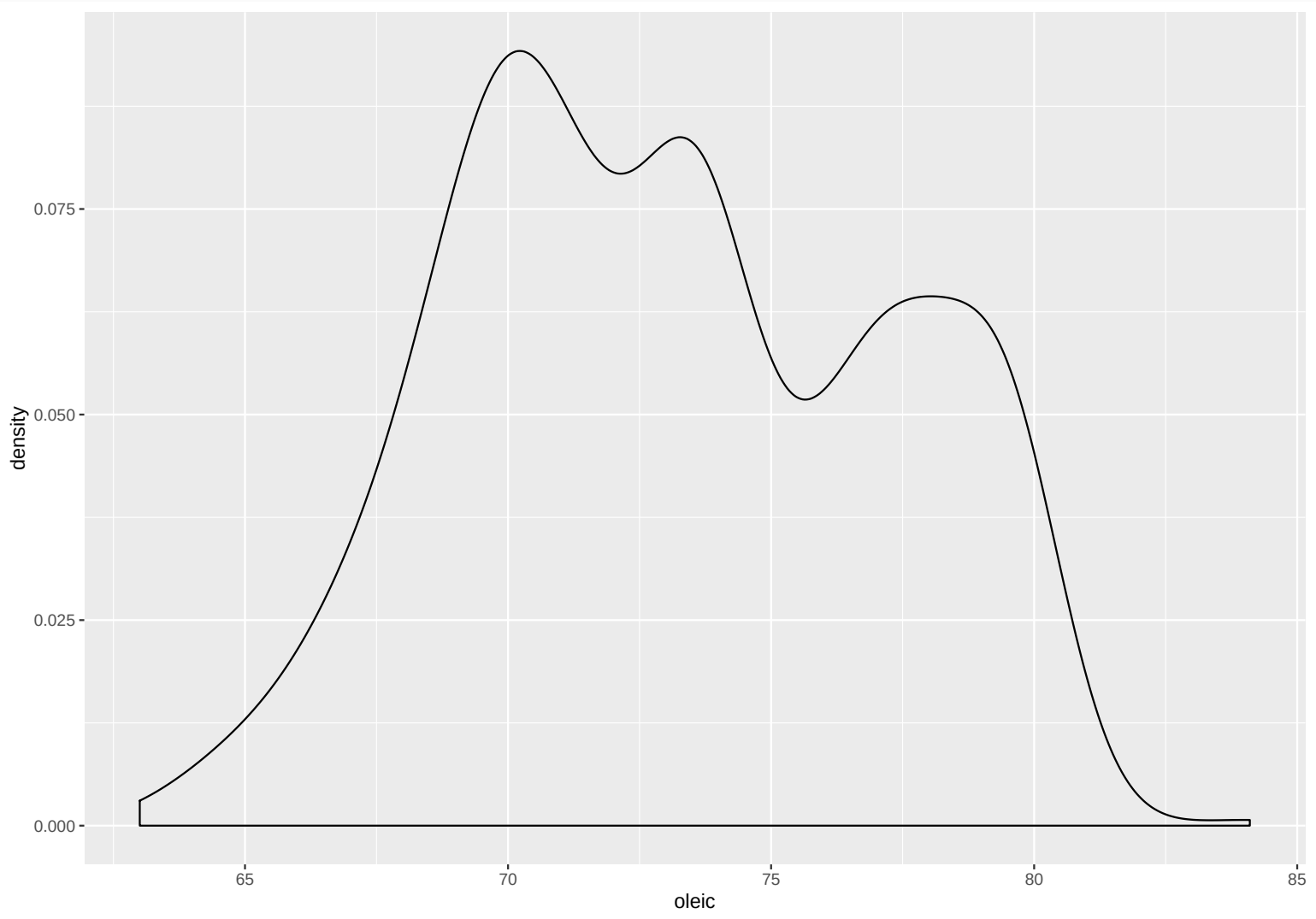
$$\hat{f}_n(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right).$$

- Many functions K can be used: gaussian, rectangular, triangular, Epanechnikov, biweight, cosine or optcosine (e.g. see Wikipedia)

Density plot ii

```
olive %>%
  ggplot(aes(oleic)) +
  geom_density()
```

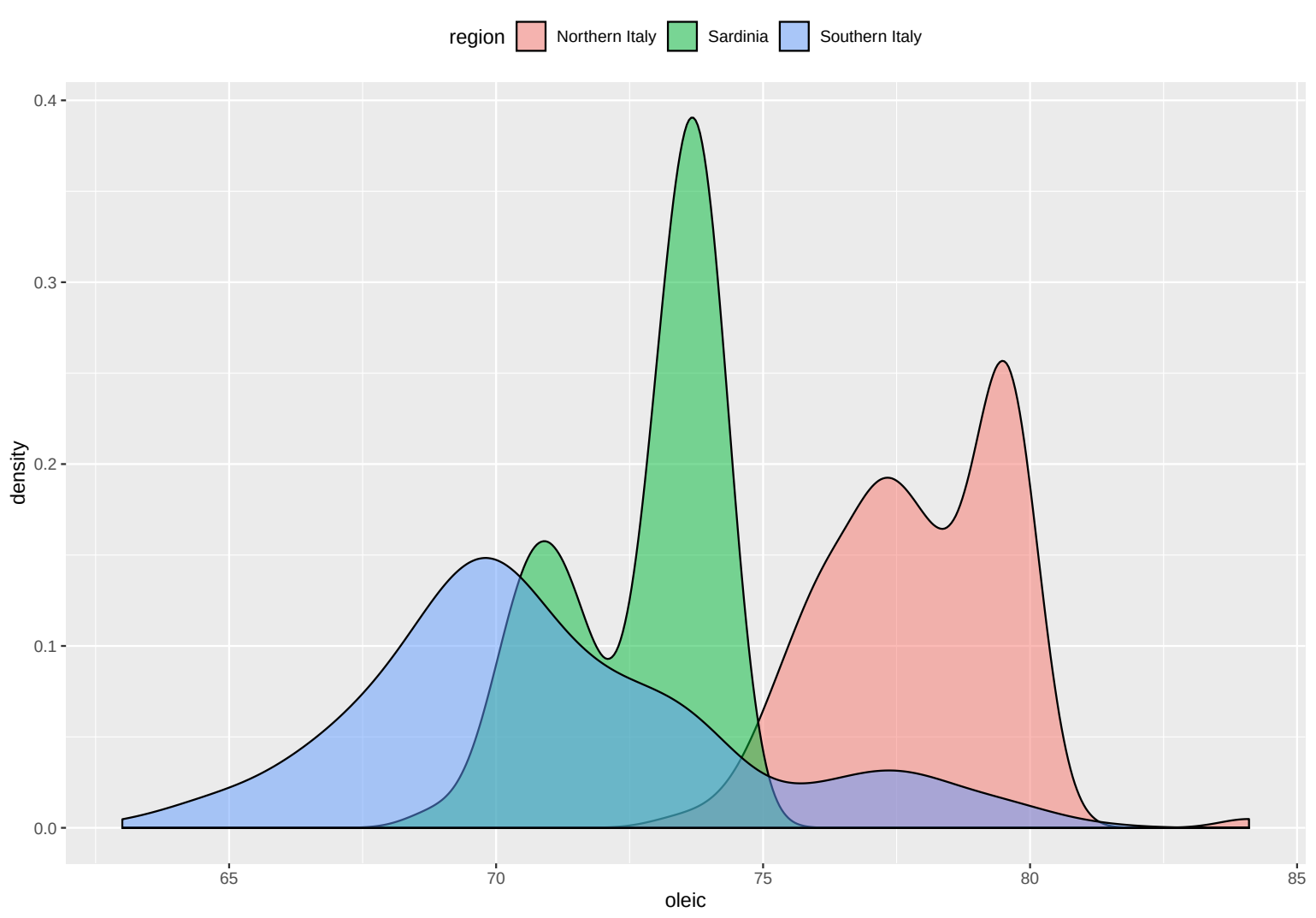
Density plot iii



Density plot iv

```
olive %>%
  ggplot(aes(oleic, fill = region)) +
  geom_density(alpha = 0.5) +
  theme(legend.position = 'top')
```

Density plot v



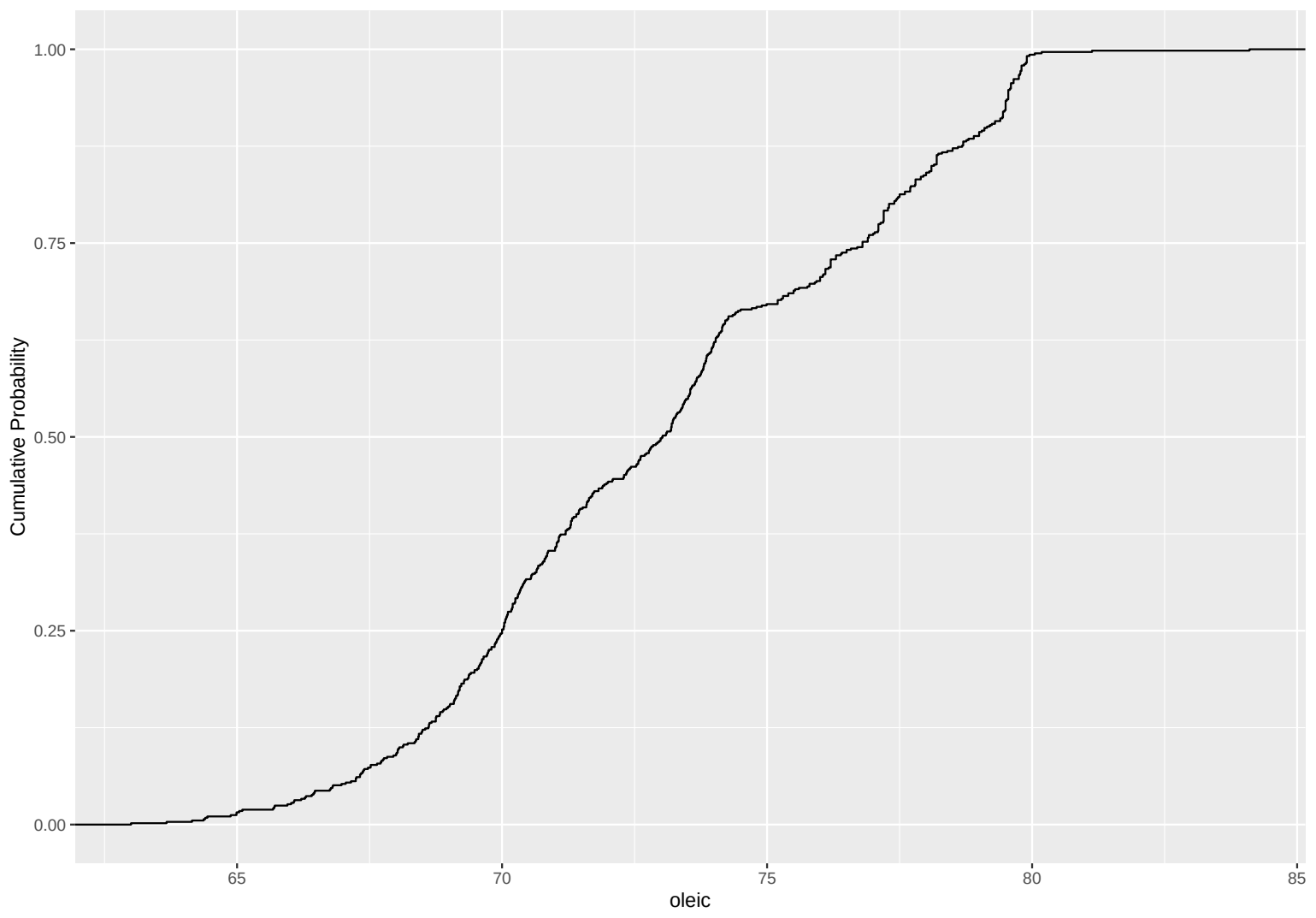
ECDF plot i

- Density plots are "smoothed histograms"
- The smoothing can hide important details, or even create artifacts
- Another way of looking at the distribution: **Empirical CDFs**
  - Easily compute/compare quantiles
  - Steepness corresponds to variance

ECDF plot ii

```
olive %>%
  ggplot(aes(oleic)) +
  stat_ecdf() +
  ylab("Cumulative Probability")
```

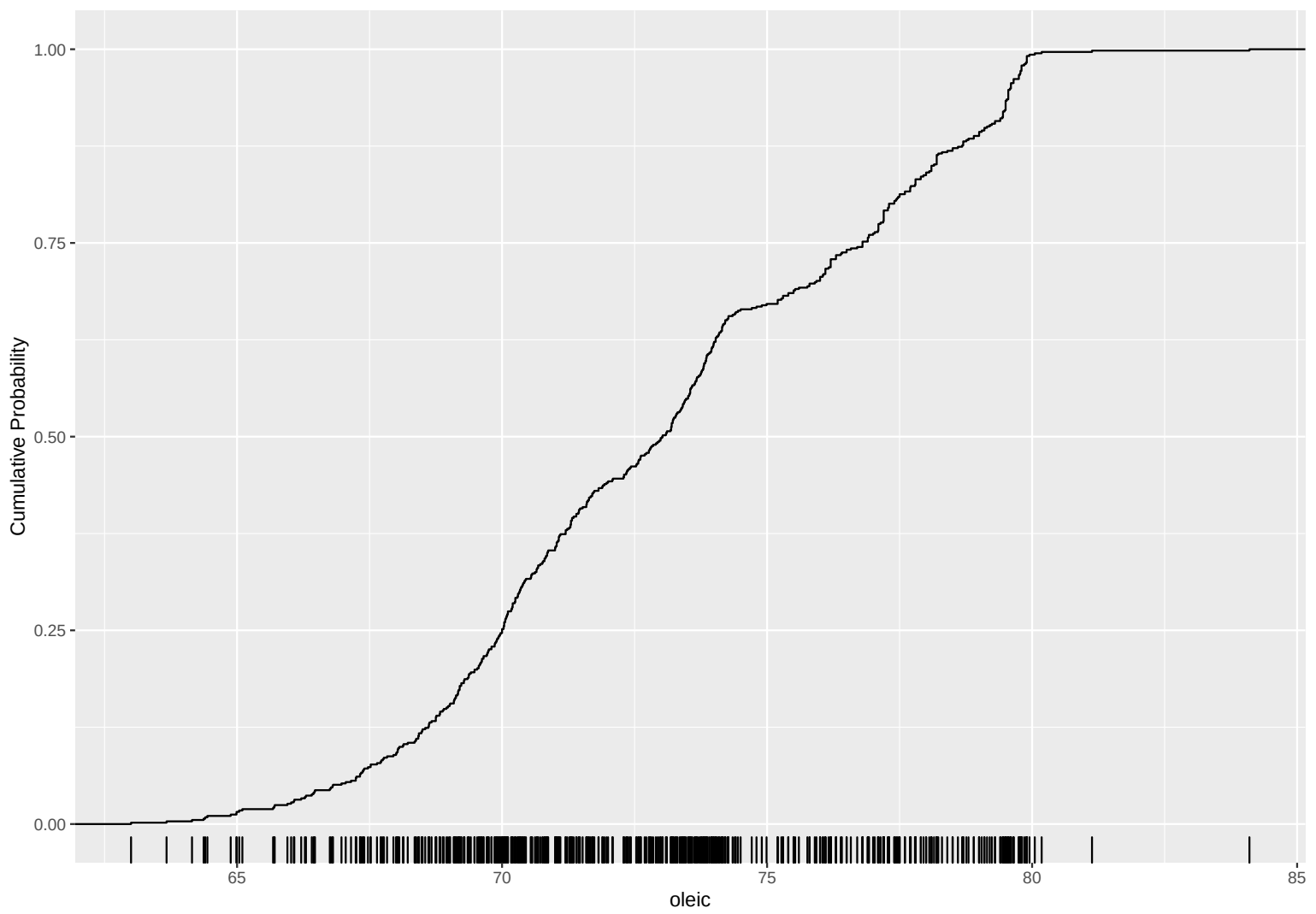
ECDF plot iii



ECDF plot iv

```
# You can add a "rug"
olive %>%
  ggplot(aes(oleic)) +
  stat_ecdf() +
  geom_rug(sides = "b") +
  ylab("Cumulative Probability")
```

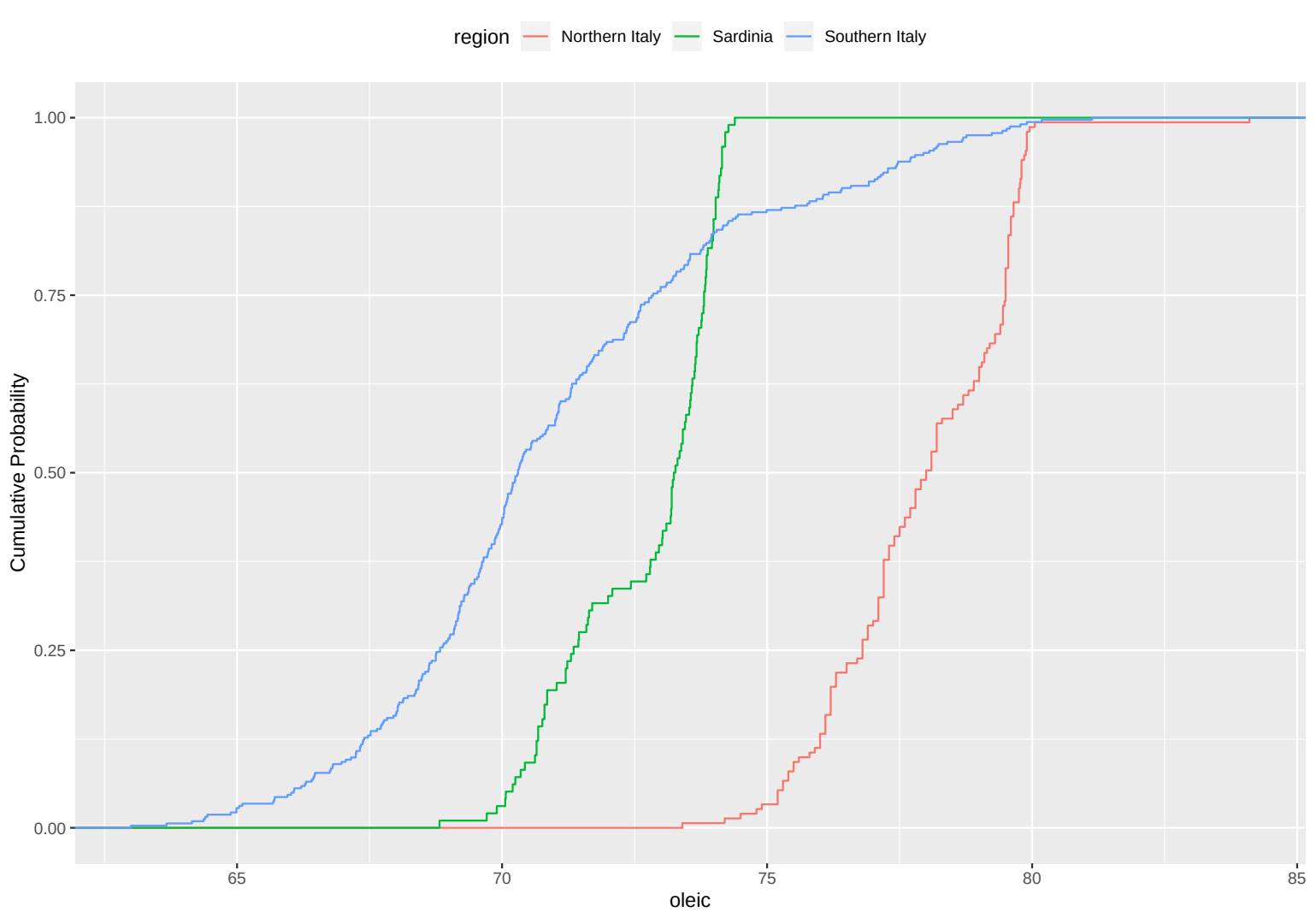
ECDF plot v


1.00
0.75
0.50
0.25
0.00
Cumulative Probability
65
70
75
80
85
oleic

ECDF plot vi

```
olive %>%
  ggplot(aes(oleic, colour = region)) +
  stat_ecdf() +
  ylab("Cumulative Probability") +
  theme(legend.position = 'top')
```

ECDF plot vii



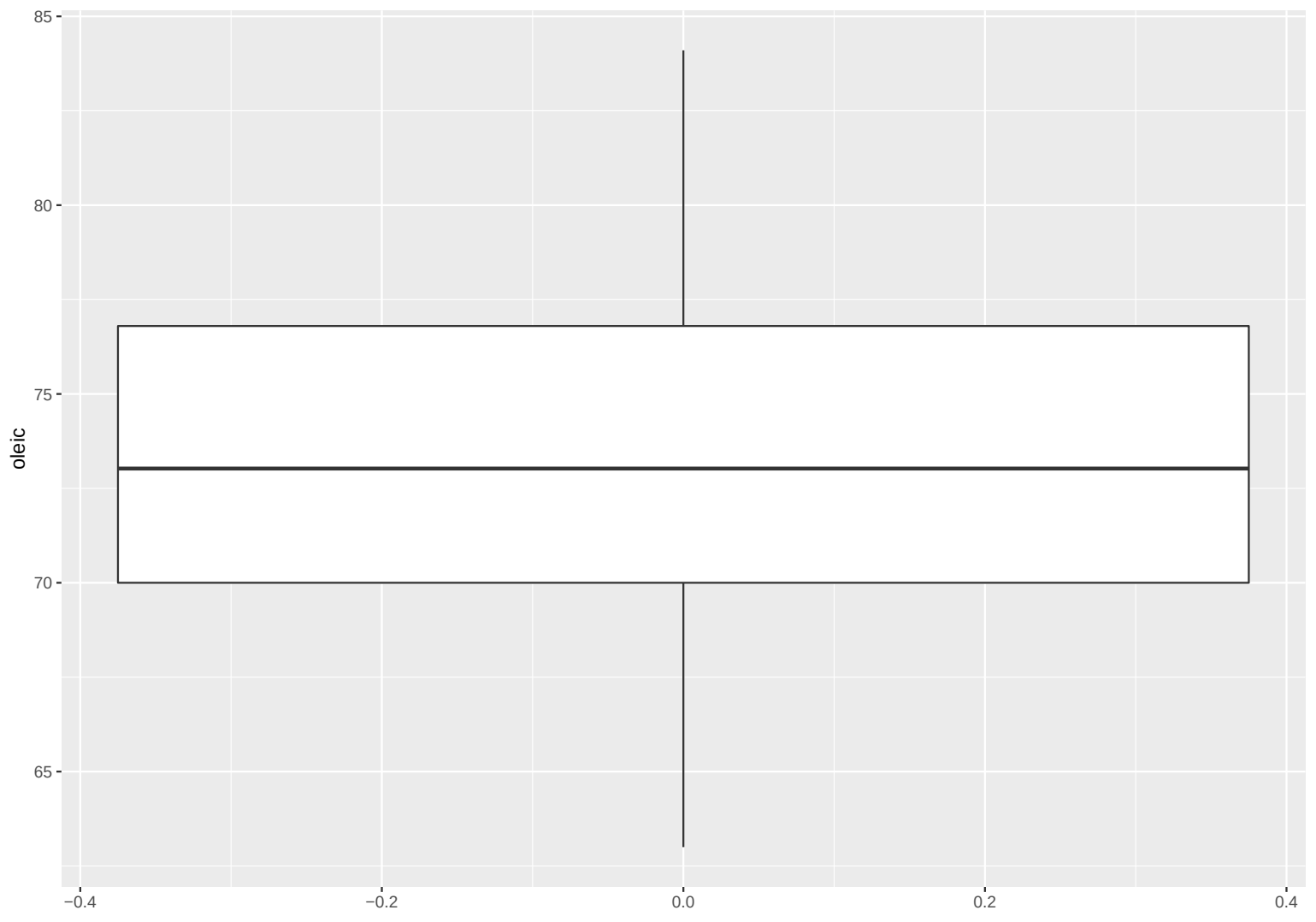
Boxplot i

- Box plots are a simple way to display important quantiles and identify outliers
- Components (per Tukey):
  - A box delimiting the first and third quartile;
  - A line indicating the median;
  - Whiskers corresponding to the lowest datum still within 1.5 IQR of the lower quartile, and the highest datum still within 1.5 IQR of the upper quartile;
  - Any datum that falls outside the whiskers is considered a (potential) outlier.

Boxplot ii

```
olive %>%
  ggplot(aes(y = oleic)) +
  geom_boxplot(x = 0)
```

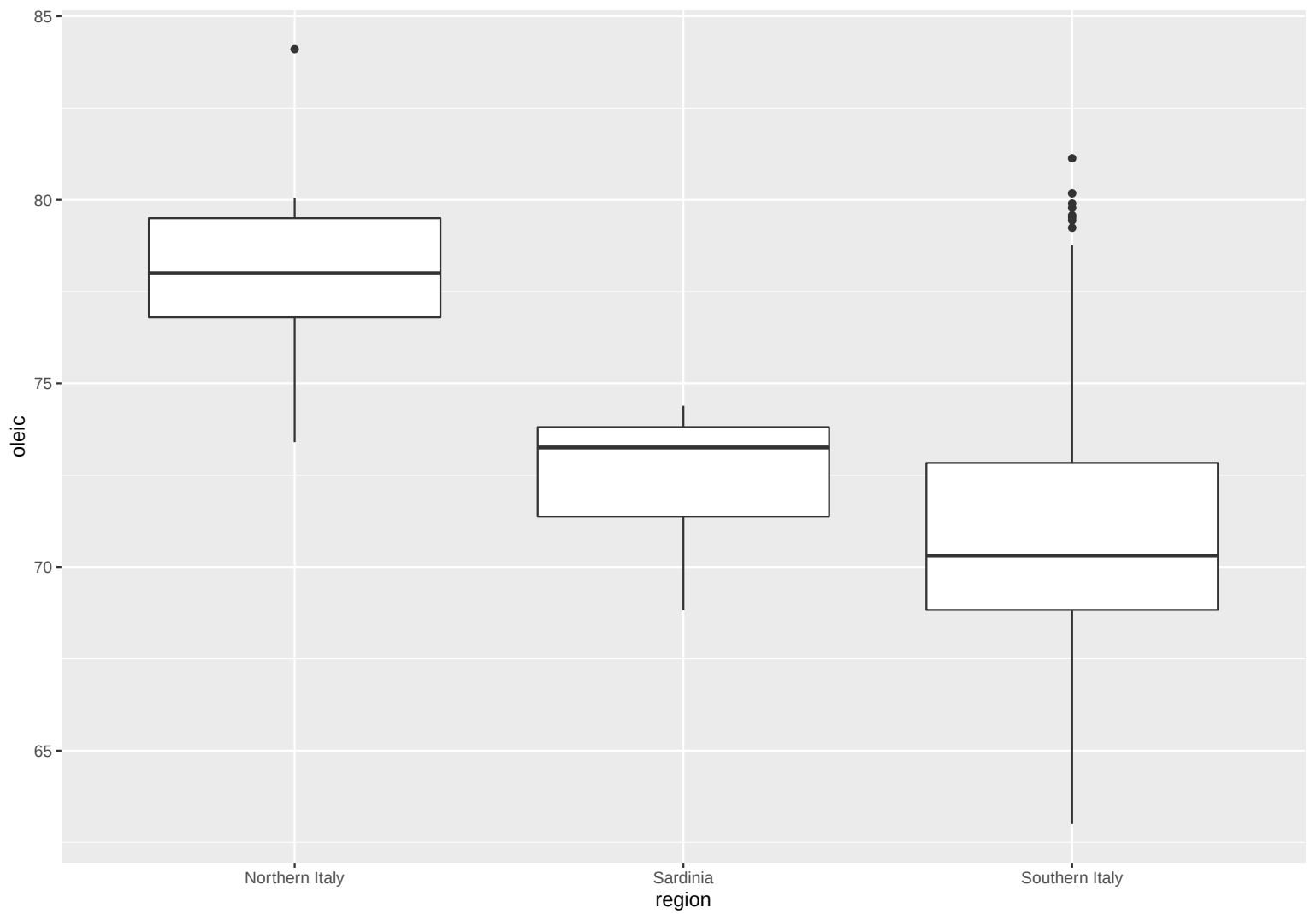
Boxplot iii



Boxplot iv

```
olive %>%
  ggplot(aes(x = region, y = oleic)) +
  geom_boxplot()
```

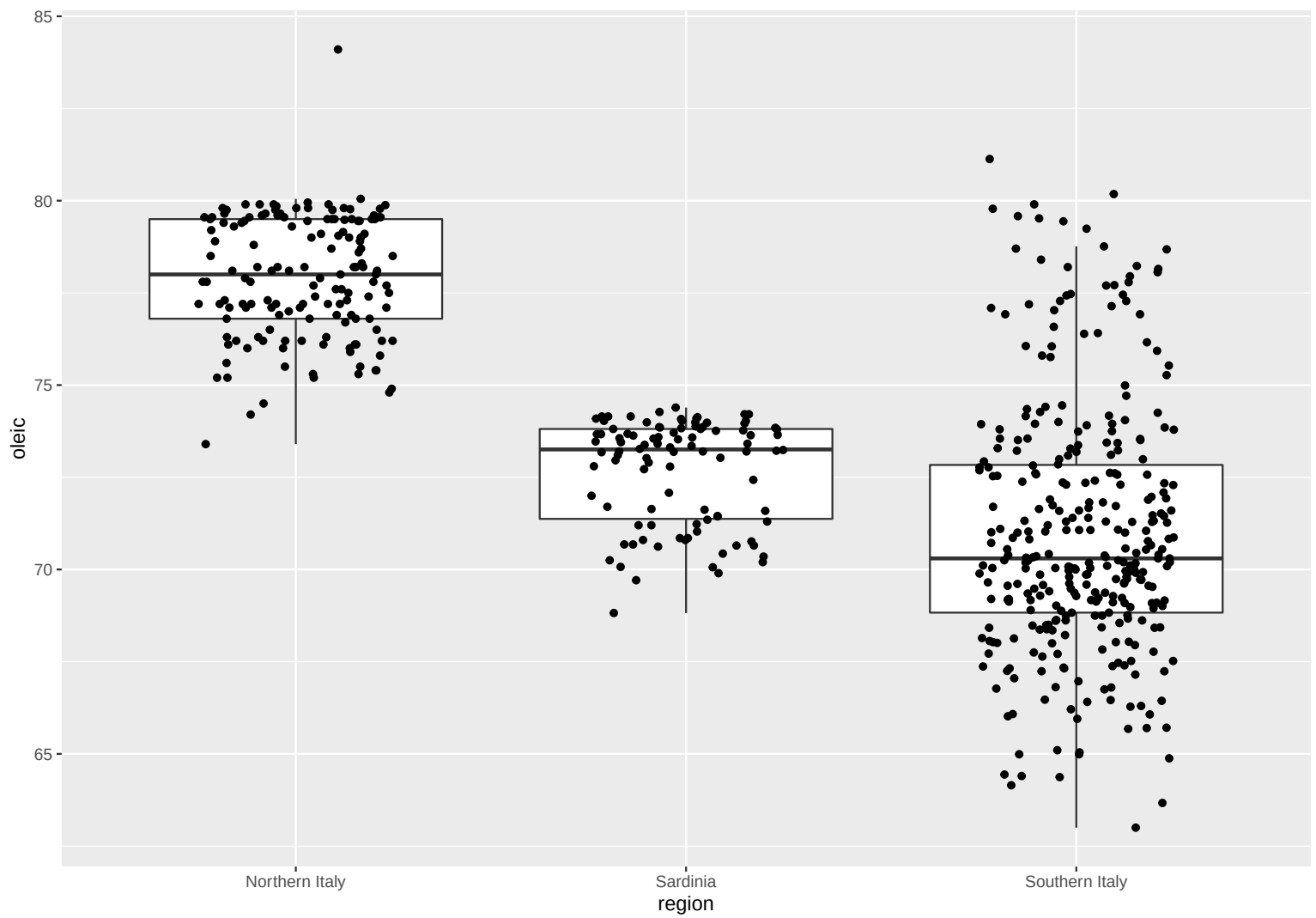
Boxplot v


85
80
75
70
65
oleic
Northern Italy
Sardinia
Southern Italy
region

Boxplot vi

```
# Add all points on top of boxplots
# Note: need to remove outliers or you will get
#       duplicates
olive %>%
  ggplot(aes(x = region, y = oleic)) +
  geom_boxplot(outlier.colour = NA) +
  geom_jitter(width = 0.25, height = 0)
```

Boxplot vii


85
80
75
70
65
oleic
Northern Italy
Sardinia
Southern Italy
region

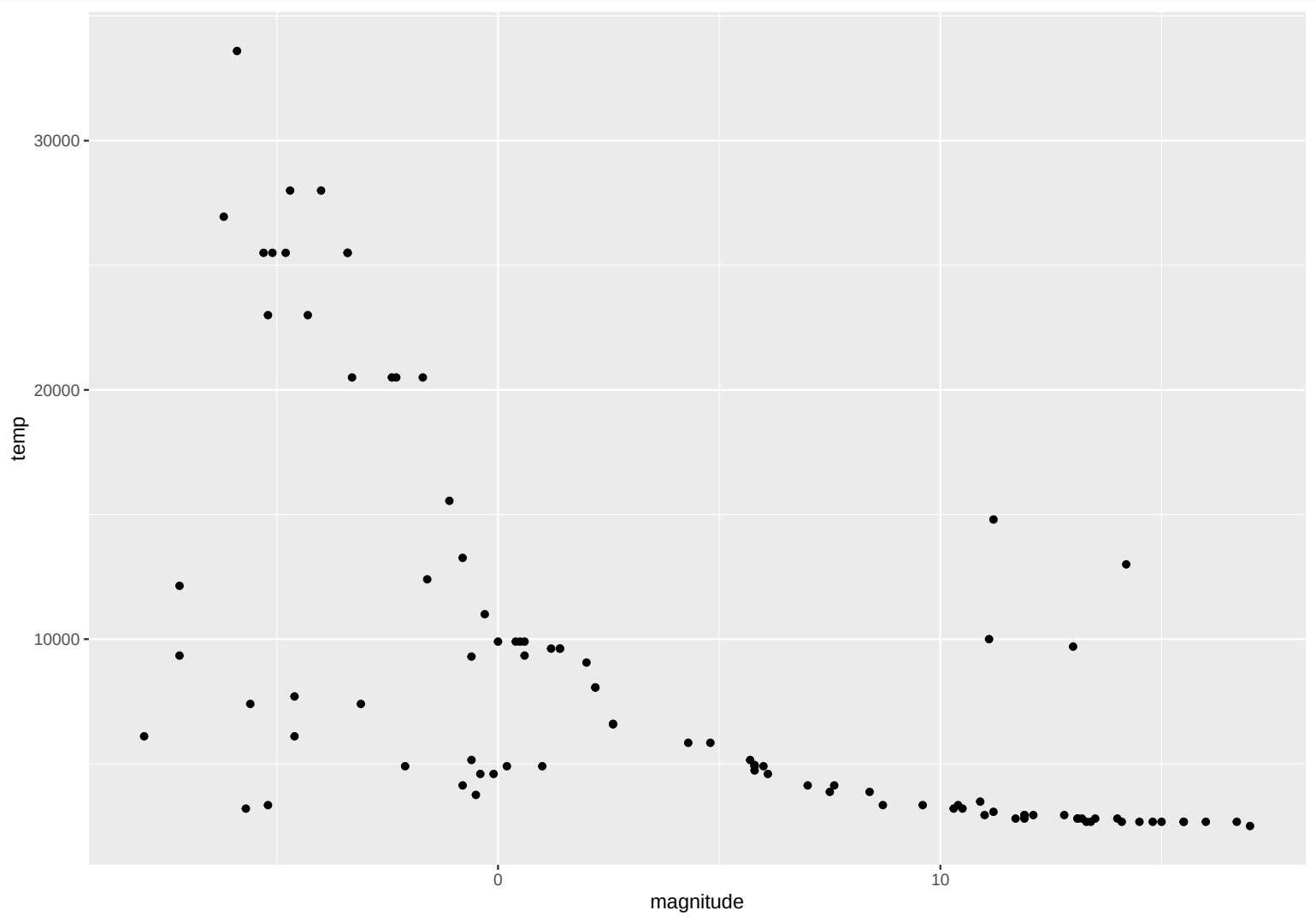
Bivariate plots

Scatter plot i

- The plots above displayed information on a single variable at a time.
- The simplest way to represent the relationship between two variables is a *scatter plot*.
- Technically still possible with three variables, but typically more difficult to read.

```
stars %>%
  ggplot(aes(magnitude, temp)) +
  geom_point()
```

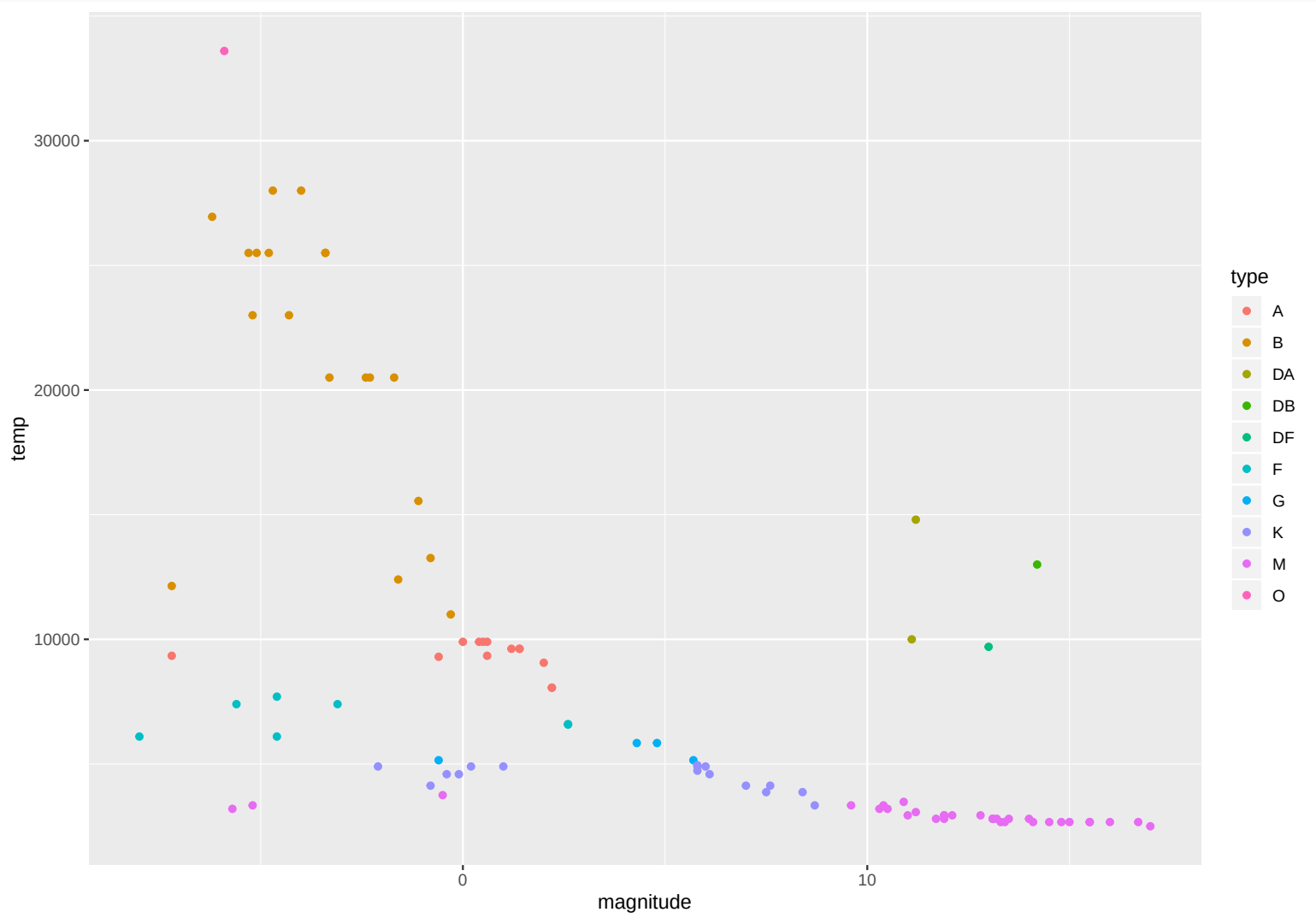
Scatter plot ii



Scatter plot iii

```
stars %>%
  ggplot(aes(magnitude, temp)) +
  geom_point(aes(colour = type))
```

Scatter plot iv

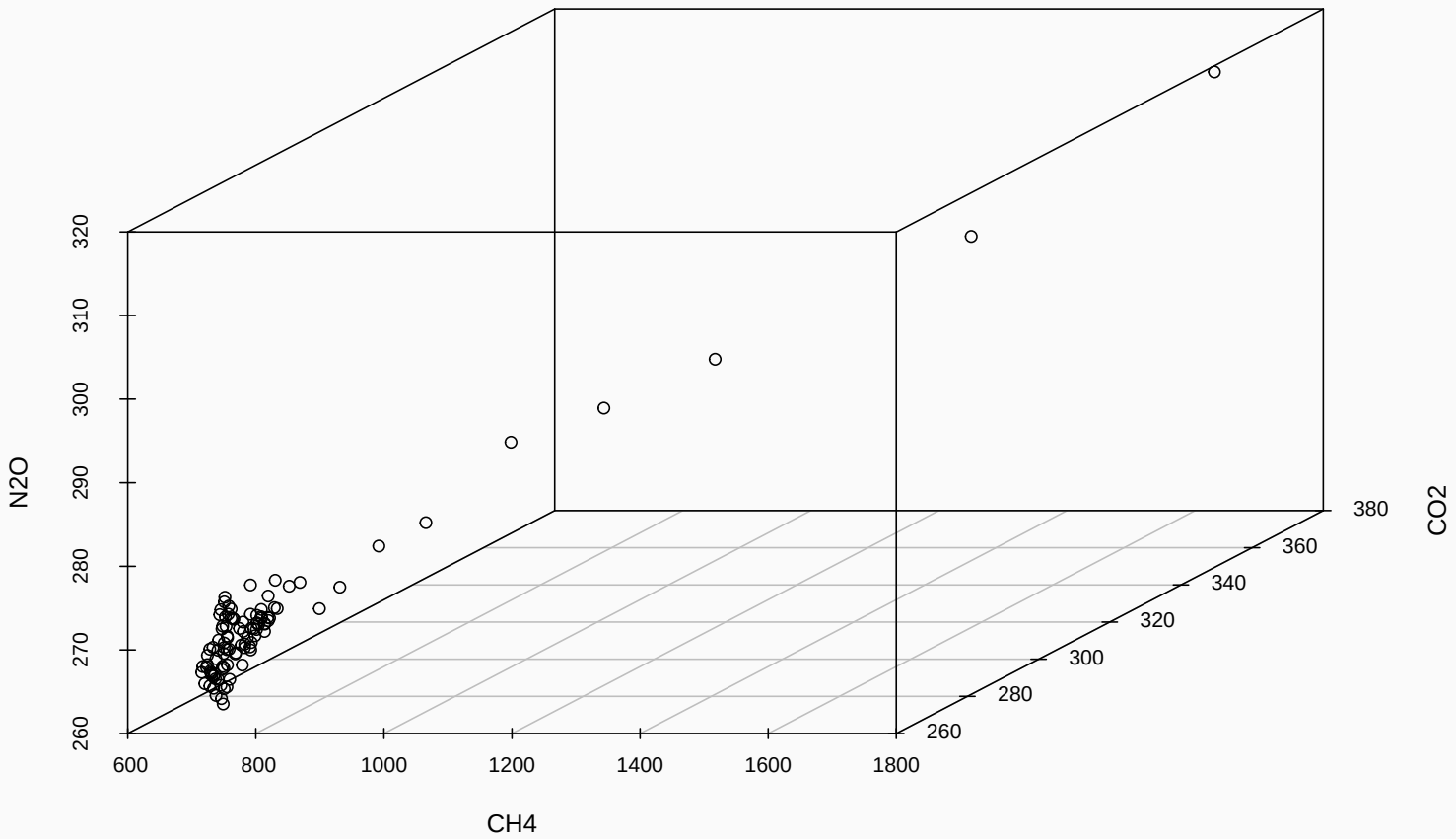


Scatter plot v

```
library(scatterplot3d)

greenhouse_gases %>%
  spread(gas, concentration) %>%
  with(scatterplot3d(CH4,    # x axis
                     CO2,    # y axis
                     N2O     # z axis
))
```

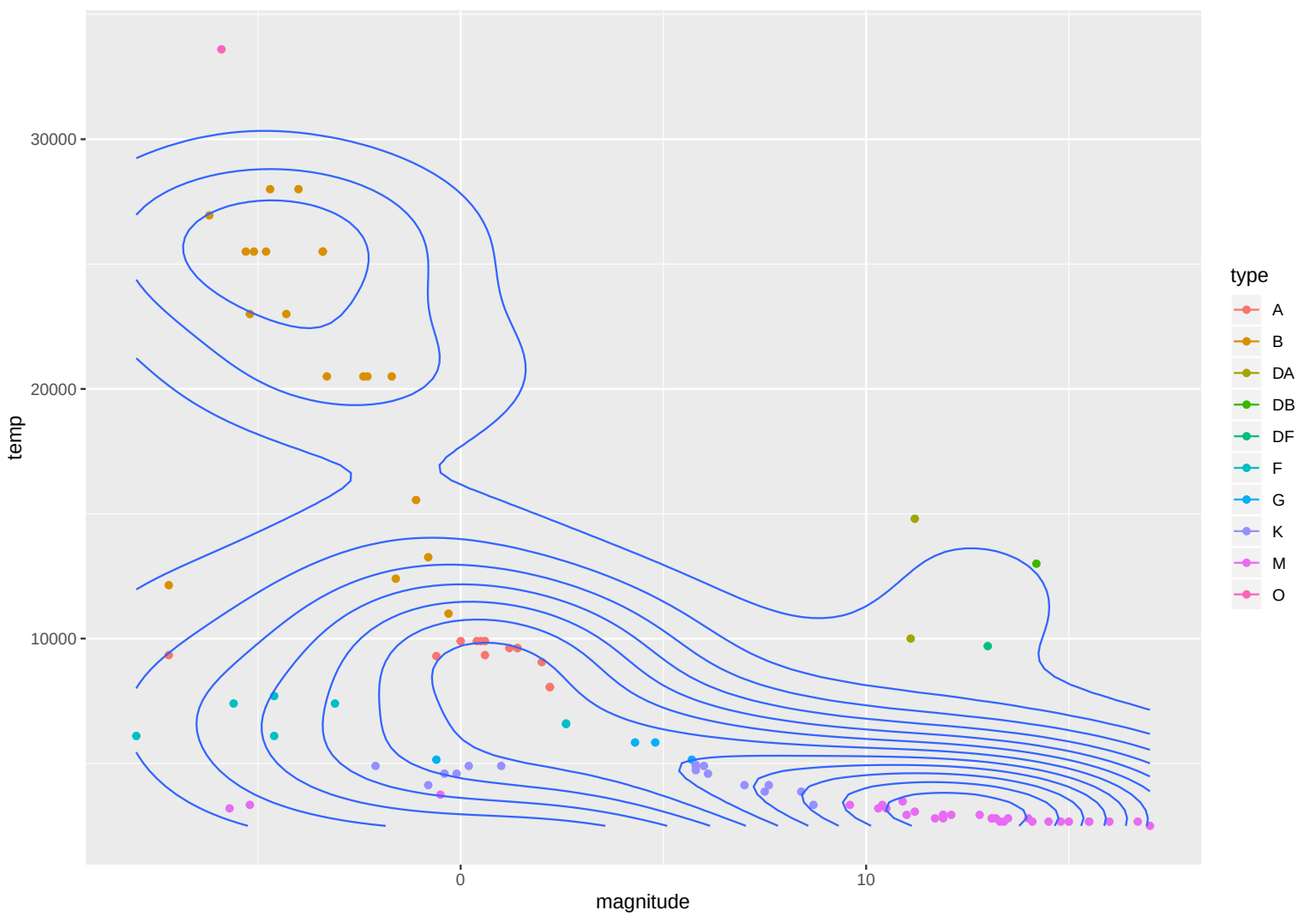
Scatter plot vi



Bivariate density plot i

```
stars %>%
  ggplot(aes(magnitude, temp)) +
  geom_point(aes(colour = type)) +
  geom_density_2d()
```

Bivariate density plot ii



Bagplot i

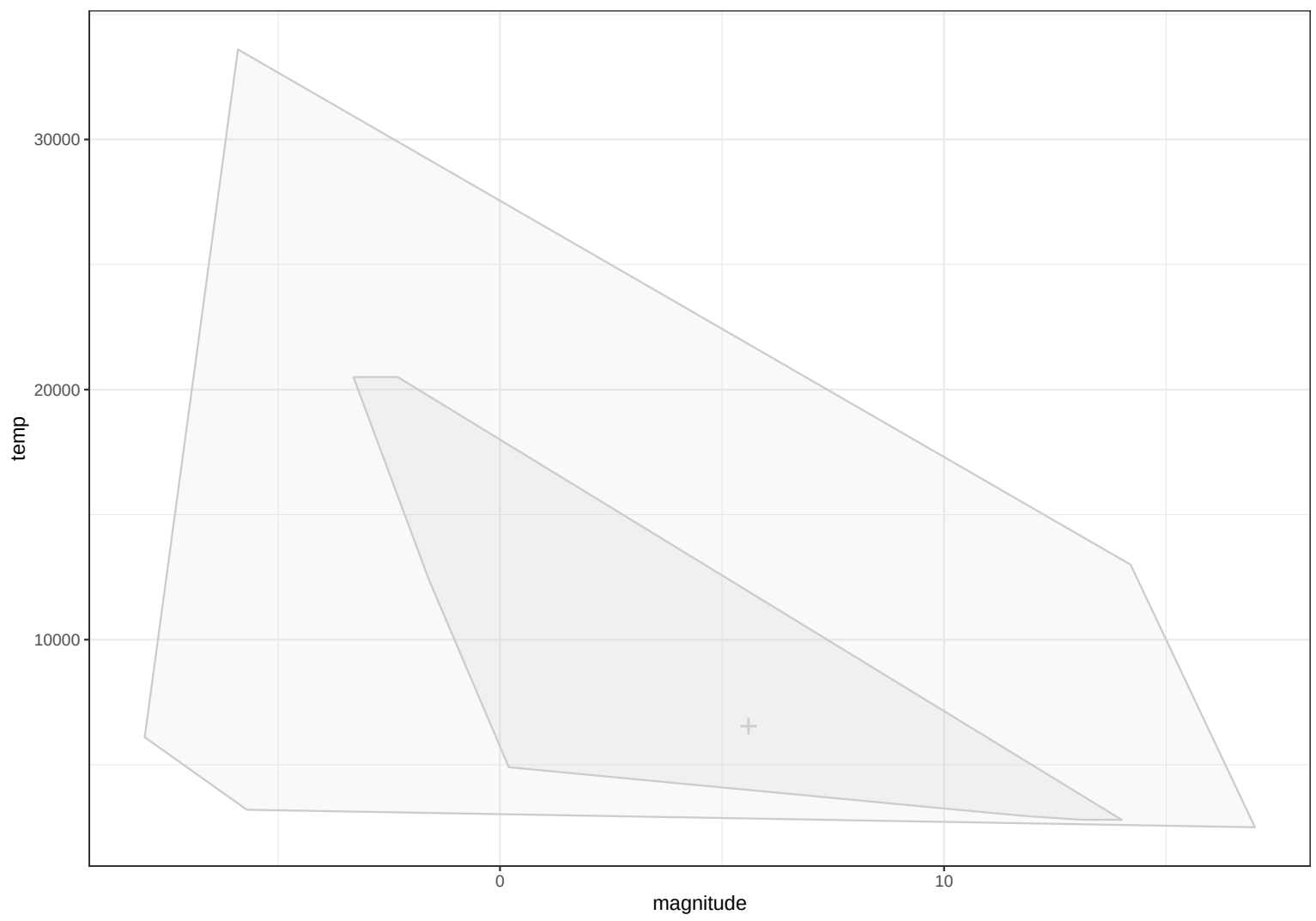
- Introduced in 1999 by Rousseuw et al. as a bivariate generalization of Tukey's boxplot.
- Help visualize location, spread, skewness, and identify potential outliers.
- Components (details omitted):
  - The *bag*, a polygon "at the center of the data cloud" that contains at most 50% of the data points.
  - The *fence*, corresponding to an inflation of the bag (typically by a factor of 3). Observations outside the fence are potential outliers.
  - The *loop*, which is the convex hull of the non-outliers.

Bagplot ii

```
devtools::source_gist("00772ccea2dd0b0f1745",
                      filename = "000_geom_bag.r")
devtools::source_gist("00772ccea2dd0b0f1745",
                      filename = "001_bag_functions.r")

stars %>%
  ggplot(aes(magnitude, temp)) +
  geom_bag() +
  theme_bw()
```

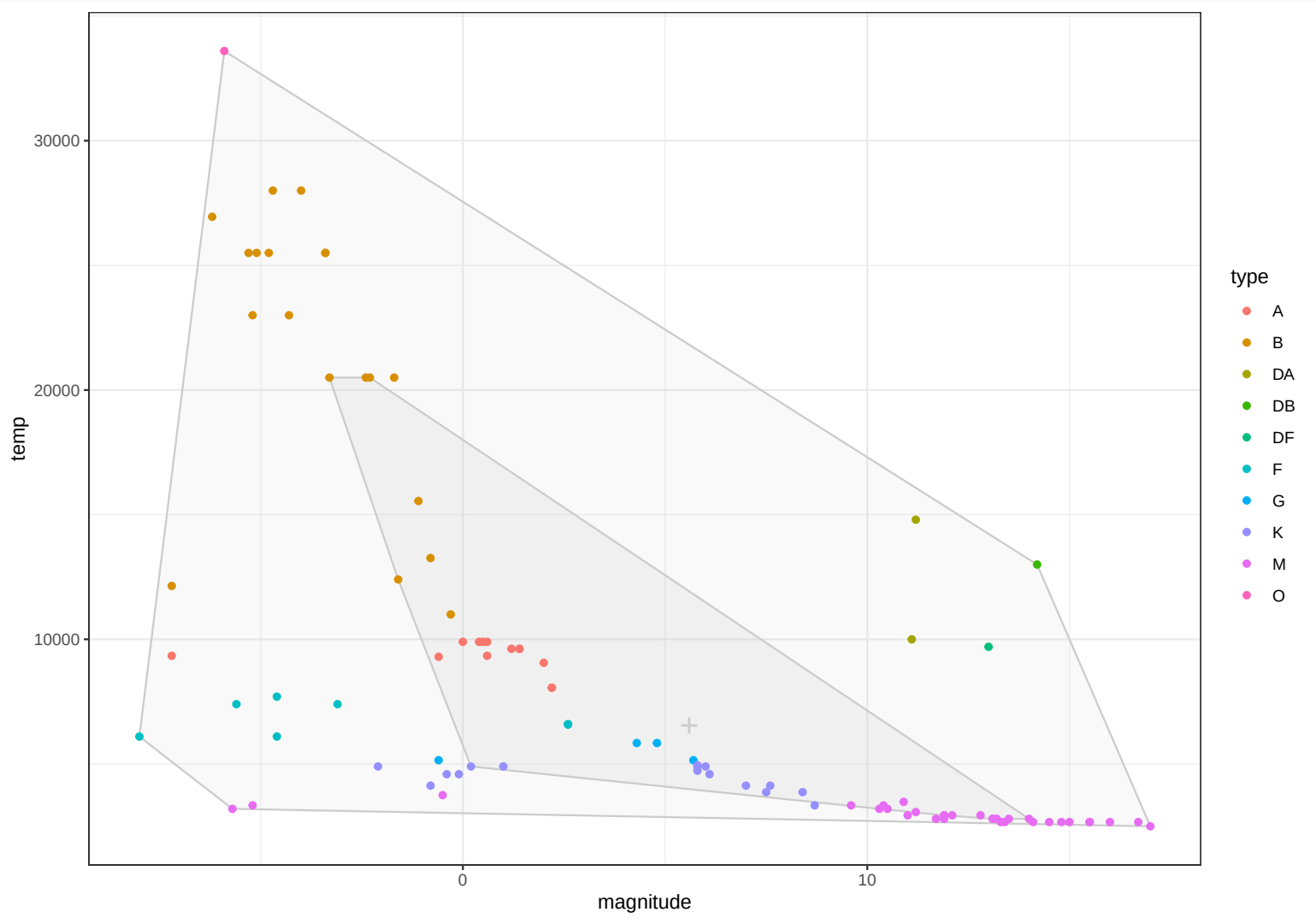
Bagplot iii



Bagplot iv

```
stars %>%
  ggplot(aes(magnitude, temp)) +
  geom_bag() +
  geom_point(aes(colour = type)) +
  theme_bw()
```

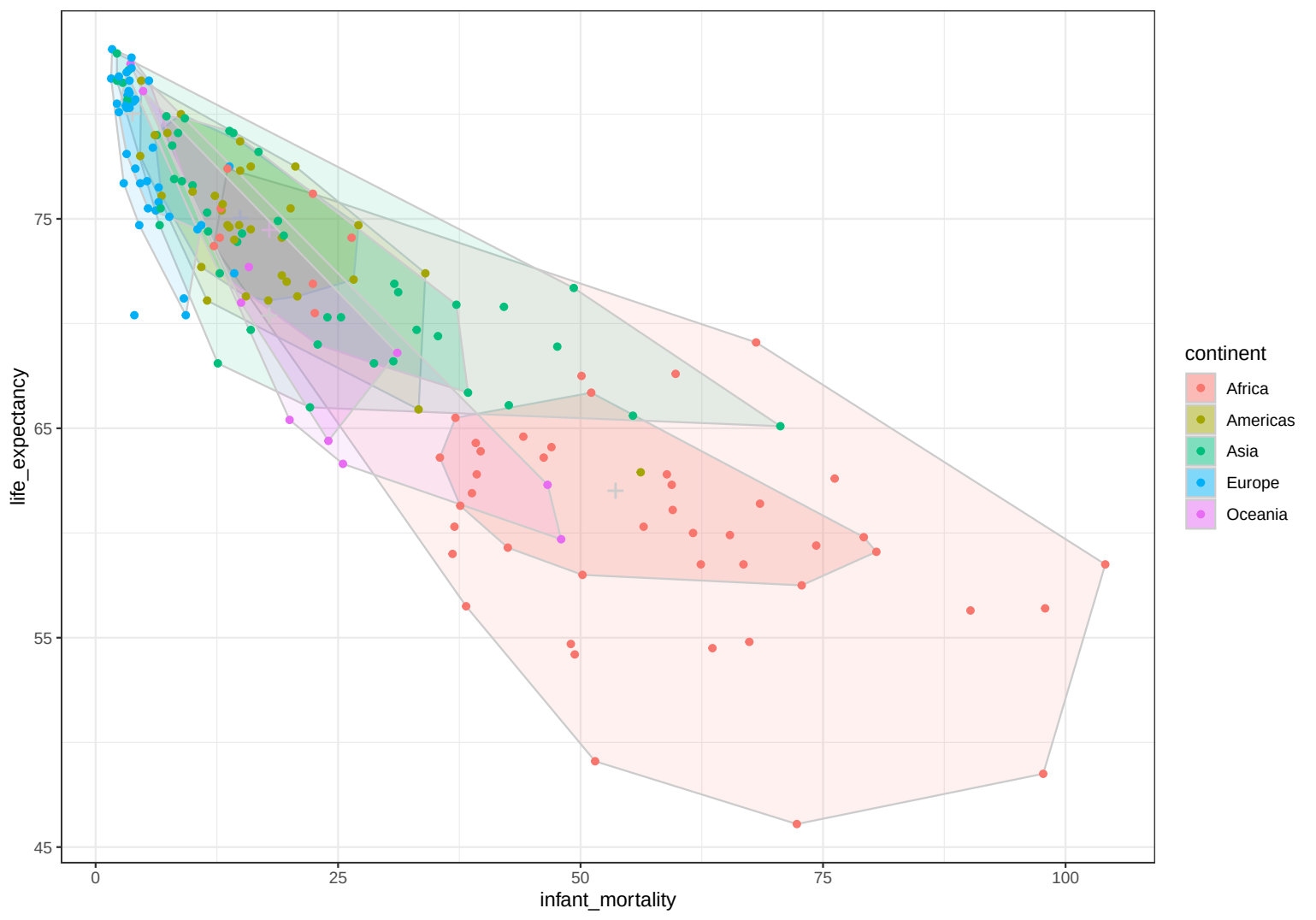
Bagplot v



Bagplot vi

```
gapminder %>%
  filter(year == 2012,
         !is.na(infant_mortality)) %>%
  ggplot(aes(infant_mortality, life_expectancy)) +
  geom_bag(aes(fill = continent)) +
  geom_point(aes(colour = continent)) +
  theme_bw()
```

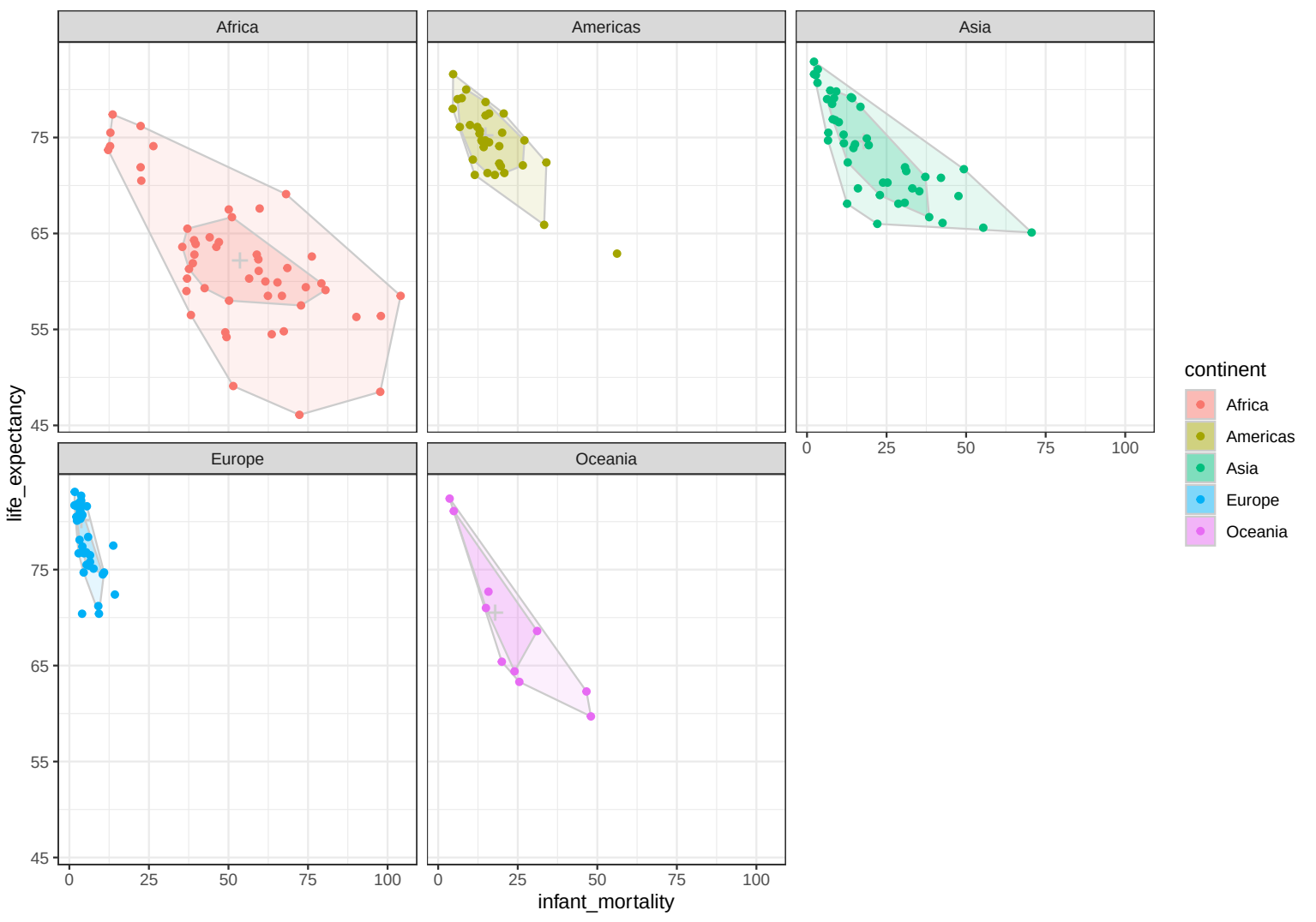
Bagplot vii


life_expectancy
75
65
55
45
0
25
50
75
100
infant_mortality
continent
Africa
Americas
Asia
Europe
Oceania

Bagplot viii

```
gapminder %>%
  filter(year == 2012,
         !is.na(infant_mortality)) %>%
  ggplot(aes(infant_mortality, life_expectancy)) +
  geom_bag(aes(fill = continent)) +
  geom_point(aes(colour = continent)) +
  facet_wrap(~continent) +
  theme_bw()
```

Bagplot ix


Africa
Americas
Asia
Europe
Oceania
life_expectancy
infant_mortality
75
65
55
45
0
25
50
75
100
continent
Africa
Americas
Asia
Europe
Oceania

Beyond two variables

Limitations

- As we saw, three-dimensional scatter plots can be hard to interpret.
- And three-dimensional bagplots would be even harder!
- Density plots can technically be constructed for any dimension
    - But as the dimension increases, its performance *decreases* rapidly
- **Solution**: We can look at each variable marginally and at each pairwise comparison.

Pairs plot i

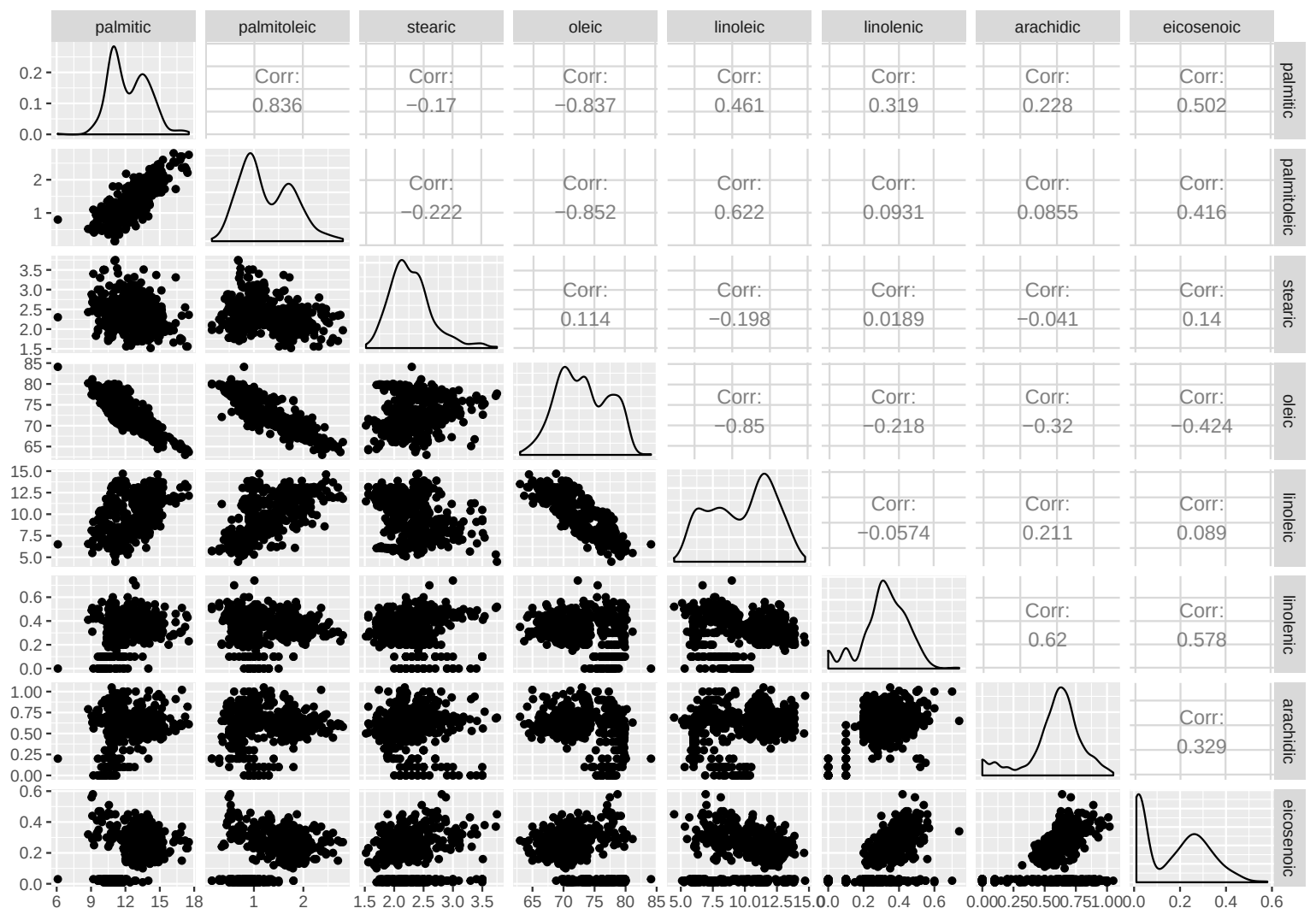
- A pairs plot arranges these univariate summaries and pairwise comparisons along a matrix.
- Each variable corresponds to both a row and a column
- Univariate summaries appear on the diagonal, and pairwise comparisons off the diagonal.
- Because of symmetry, we often see a different summary of the comparison above and below the diagonal
- I will show two packages:
  1. `GGally`
  2. `ggforce`

Pairs plot ii

```
library(GGally)

olive %>%
  dplyr::select(-region, -area) %>%
  ggpairs
```

Pairs plot iii

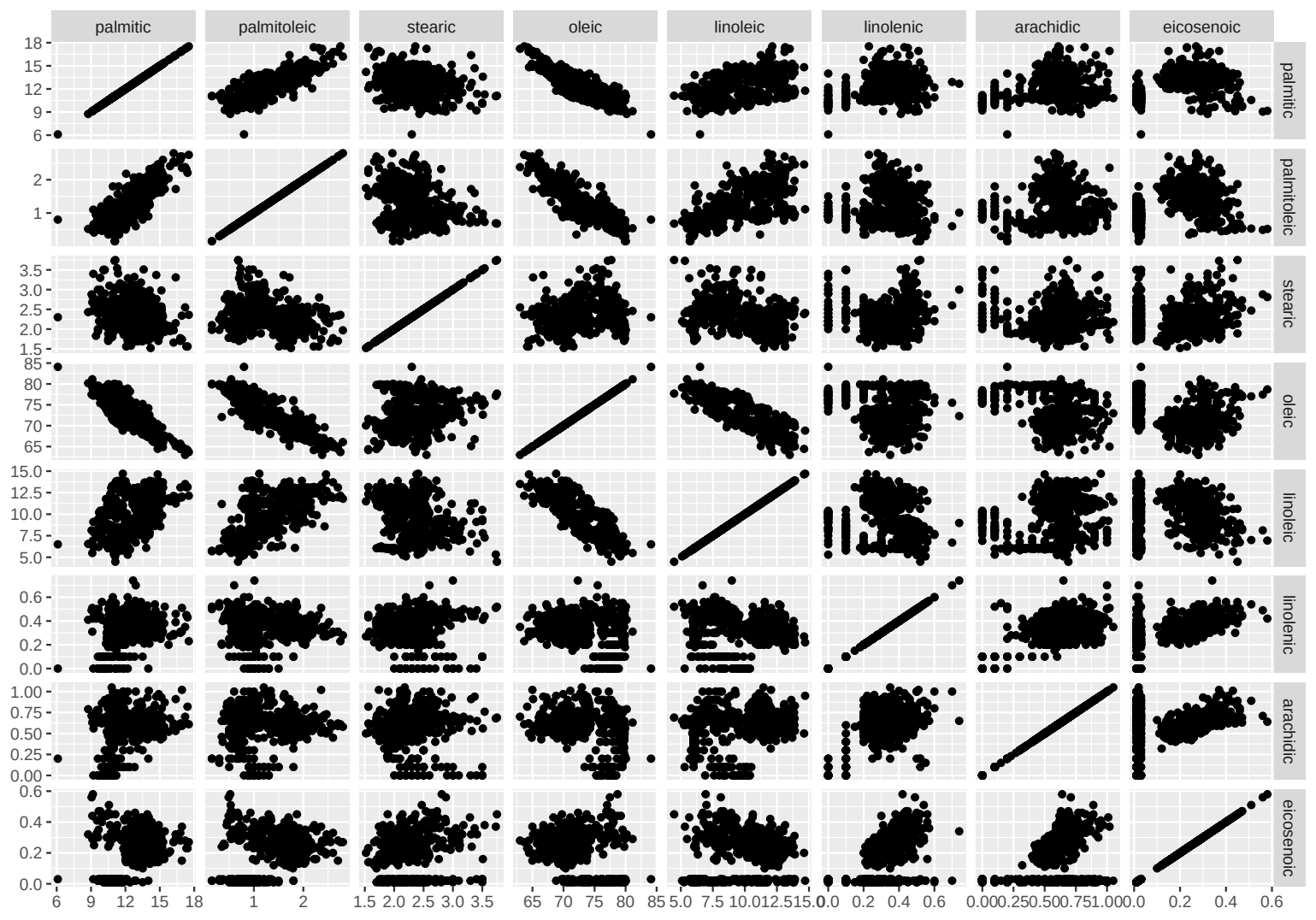


Pairs plot iv

```
library(ggforce)

olive %>%
  dplyr::select(-region, -area) %>%
  ggplot(aes(x = .panel_x, y = .panel_y)) +
  geom_point() +
  facet_matrix(vars(everything()))
```

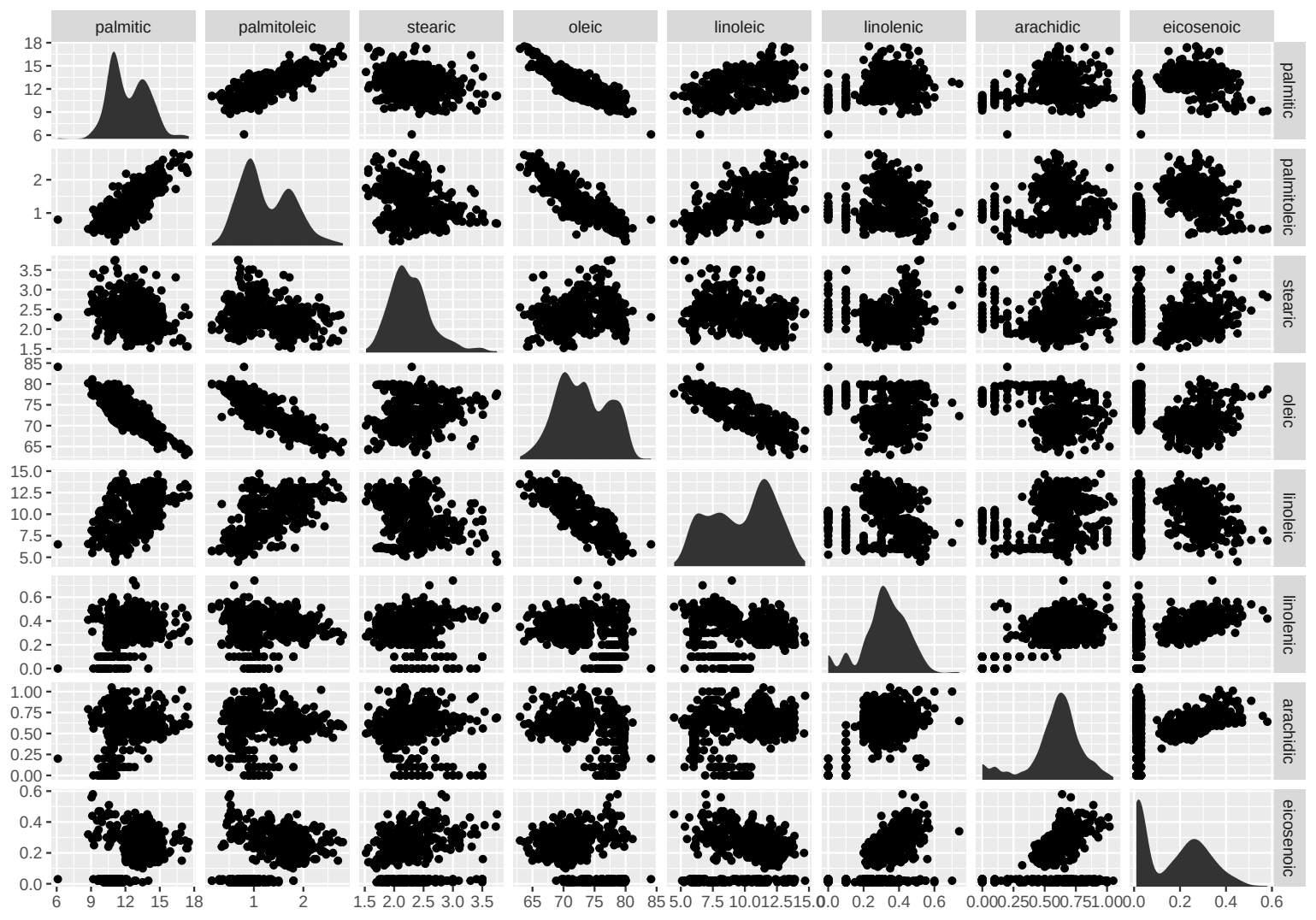
Pairs plot v


palmitic
palmitoleic
stearic
oleic
linoleic
linolenic
arachidic
eicosenoic

Pairs plot vi

```
olive %>%
  dplyr::select(-region, -area) %>%
  ggplot(aes(x = .panel_x, y = .panel_y)) +
  geom_point() +
  geom_autodensity() +
  facet_matrix(vars(everything()),
               layer.diag = 2)
```

Pairs plot vii



Pairs plot viii

```
olive %>%
  dplyr::select(-region, -area) %>%
  ggplot(aes(x = .panel_x, y = .panel_y)) +
  geom_point() +
  geom_autodensity() +
  geom_density2d() +
  facet_matrix(vars(everything()),
               layer.diag = 2,
               layer.upper = 3)
```

Pairs plot ix