

Data Visualization

Max Turgeon

DATA 2010—Tools and Techniques for Data Science

Lecture Objectives

- Identify the main types of data visualization
- Contrast their strengths and weaknesses

Motivation

- Summary statistics are useful in doing quick comparisons.
 - Or even statistical inference
- Data visualizations are an effective way of sharing *a lot* of information about a dataset.
- In this slide deck, we'll focus on the main types of data visualizations; in the next one, we'll discuss important principles for **effective** visualization.

Main principles

Why would we want to visualize data?

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

Histogram i

- A **histogram** represents the frequency of observations occurring in certain bins.
 - Most software will choose default bins, but you can always change them.
- It is useful for displaying continuous data, and comparing its distribution across subgroups.

```
library(tidyverse)
```

```
library(dslabs)
```

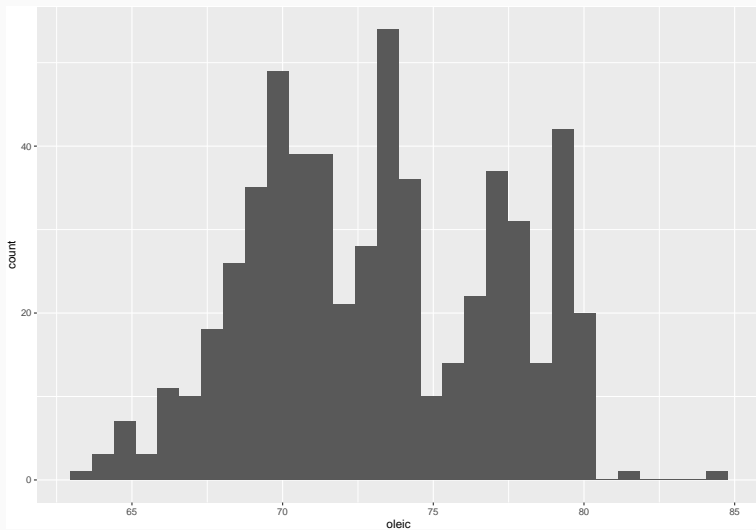
```
dim(olive)
```

Histogram ii

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

```
# Create histogram for oleic acid  
ggplot(olive,  
       aes(x = oleic)) +  
  geom_histogram()
```

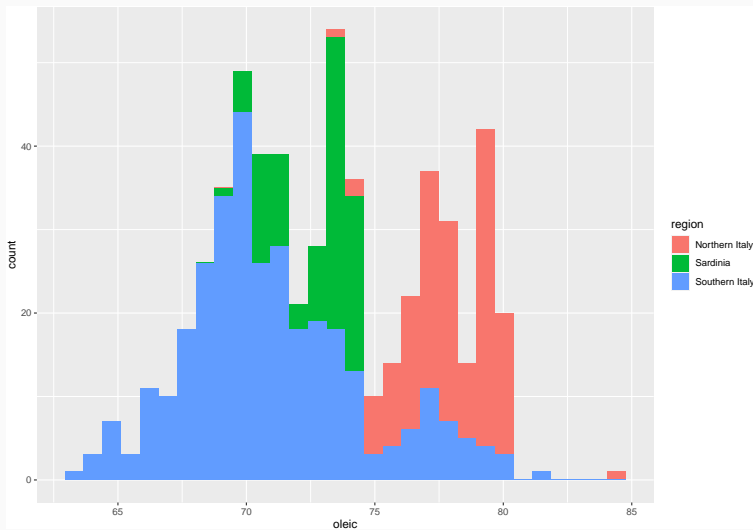
Histogram iii



Histogram iv

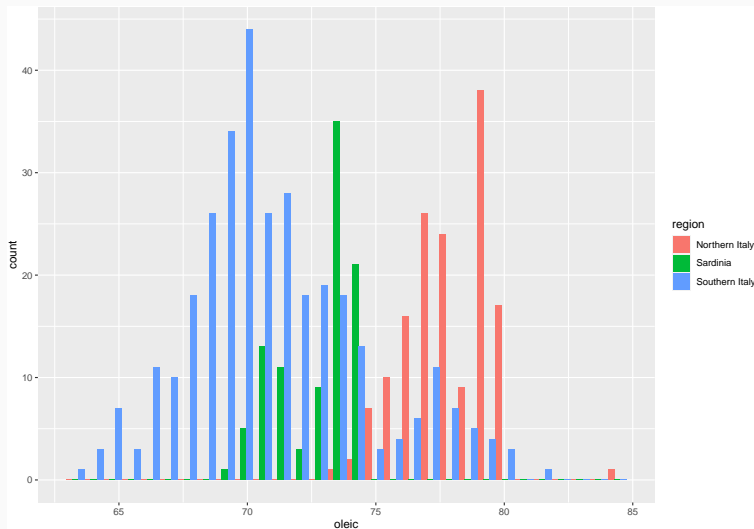
```
# Look at distribution by region  
ggplot(olive,  
       aes(x = oleic, fill = region)) +  
  geom_histogram()
```


Histogram v



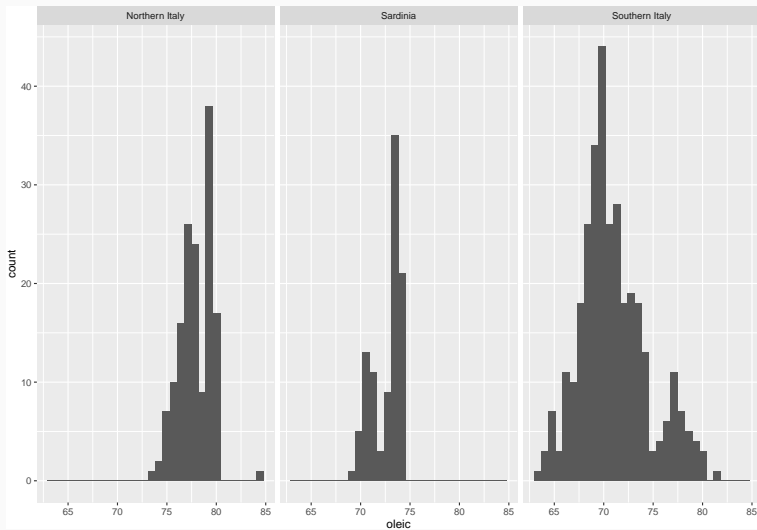
```
# Dodge instead of stack
ggplot(olive,
       aes(x = oleic, fill = region)) +
  geom_histogram(position = "dodge")
```

Histogram vii



```
# Or with facets  
ggplot(olive,  
       aes(x = oleic)) +  
  geom_histogram() +  
  facet_grid(. ~ region)
```

Histogram ix



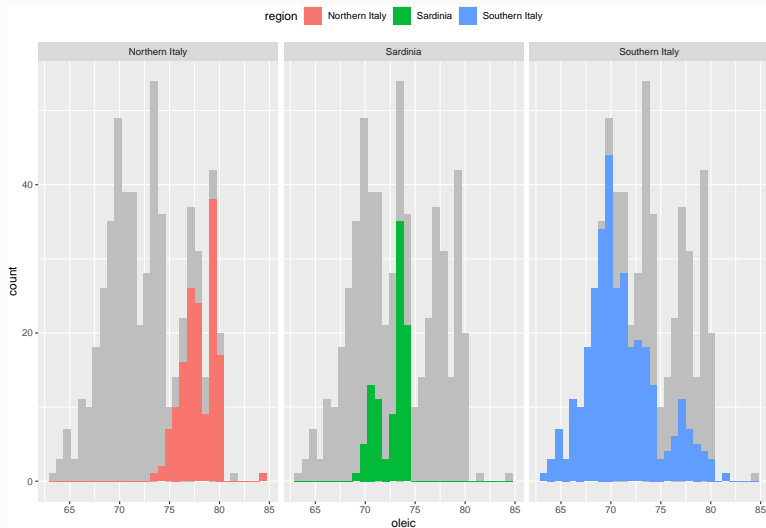
Histogram—Summary

- **Histograms** help visualize the distribution of a single variable.
 - It bins data and displays the counts in each bin
 - But large bins can hide important features, while small bins can create artifacts.
- **ggplot** takes a **data.frame** as input and maps variables to different features of the graph.
 - **oleic** is mapped to the **x-axis**
 - **region** is mapped to the **fill** colour.
 - **Important:** This mapping happens inside the function **aes**.
- **ggplot2** automatically takes care of choosing the colour, drawing the limits, and printing a legend.
- **facet_grid** can be used to display multiple plots together, one per value of the variable.

A more complex histogram i

```
# Create a copy of the data to serve as background
olive_bg <- select(olive, -region)
ggplot(olive, aes(x = oleic)) +
  # Start with grey background
  geom_histogram(data = olive_bg,
                 fill = 'grey') +
  # Add colour on top
  geom_histogram(aes(fill = region)) +
  facet_grid(. ~ region) +
  # Move legend to top
  theme(legend.position = 'top')
```

A more complex histogram ii



A more complex histogram iii

- **Note:** The order in which you add the layers (e.g. the two `geom_histogram`) is important!
- Try swapping the two `geom_histogram` calls and see what happens.

Exercise

Use the dataset `nba_players_19` from the package `openintro` to plot a histogram of the heights of basketball players.

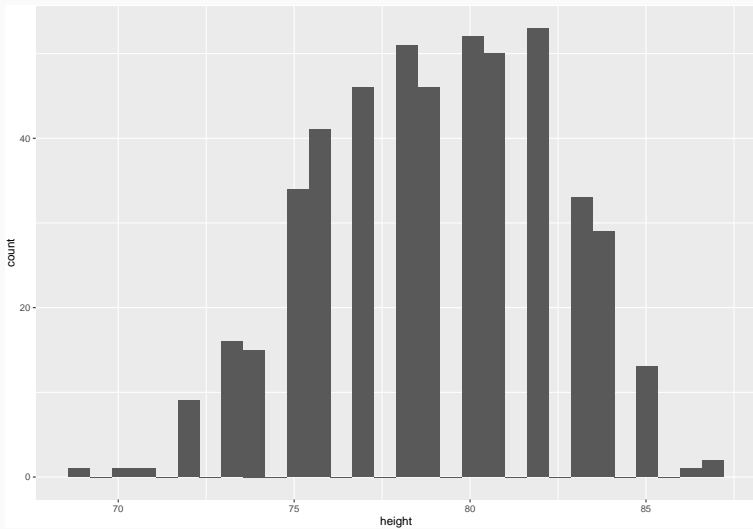
Next, use histograms to compare the height distribution of guards vs centers.

- First, we plot the overall histogram.

```
library(tidyverse)
library(openintro)

ggplot(nba_players_19, aes(height)) +
  geom_histogram()
```

Solution ii

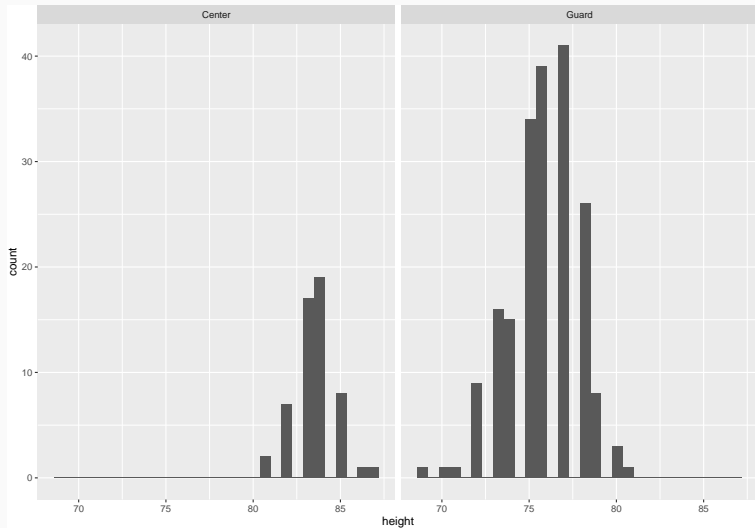


Solution iii

- Next, we need to figure out which variable encodes the position of each player.
 - You can look at the help page `?nba_players_19`.
 - You can look at `str(nba_players_19)`.
- Then we can filter using `position`.

```
nba_players_19 %>%  
  filter(position %in% c("Center", "Guard")) %>%  
  ggplot(aes(height)) +  
  geom_histogram() +  
  facet_grid(~position)
```

Solution iv

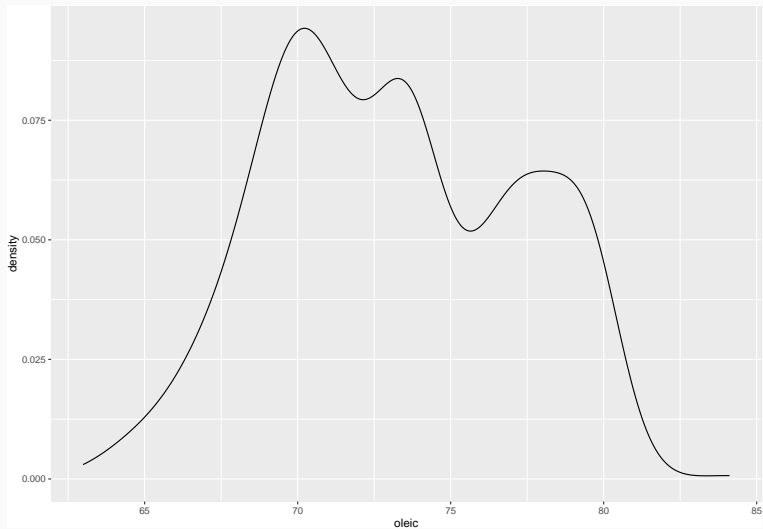


Density plot i

- **Density plots** can be thought of as *smoothed* histograms.
 - Their mathematical definition is much more involved and beyond the scope of this course.
- They can be used interchangeably with histograms.

```
ggplot(olive, aes(x = oleic)) +  
  geom_density()
```

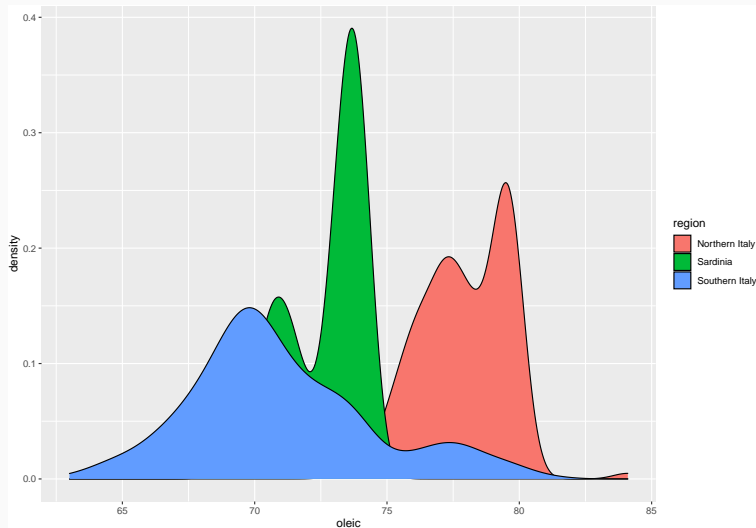
Density plot ii



Density plot iii

```
# Split by region  
ggplot(olive, aes(x = oleic,  
                  fill = region)) +  
  geom_density()
```

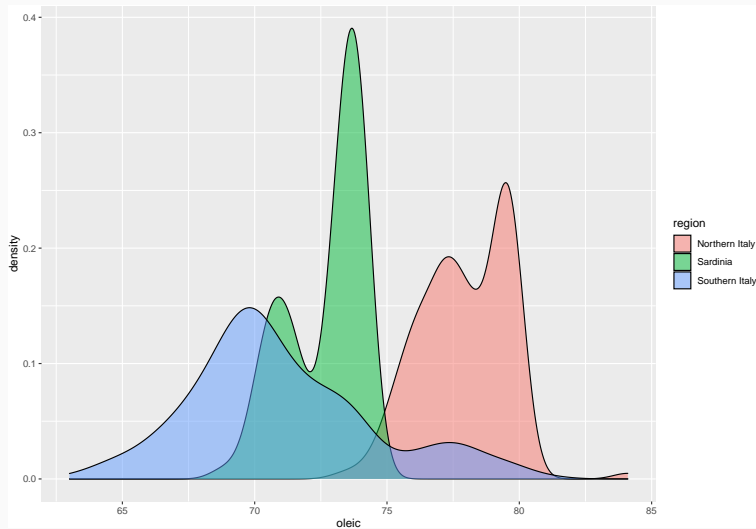
Density plot iv



Density plot v

```
# Add transparency  
ggplot(olive, aes(x = oleic,  
                  fill = region)) +  
  geom_density(alpha = 0.5)
```

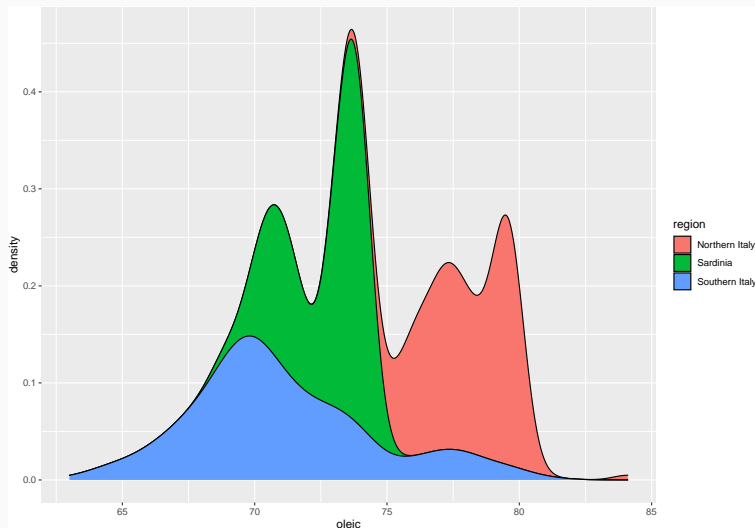
Density plot vi



Density plot vii

```
# Alternative: stacked density plots
ggplot(olive, aes(x = oleic,
                  fill = region)) +
  geom_density(position = "stack")
```

Density plot viii



Density plot—Summary

- **Density plots** can be thought of as *smoothed* histograms.
 - There is a parameter (**adjust**) controlling the level of smoothness: too large and it will hide important features; too small and it may create artifacts.
- We used a different *geom* to create the plot.
 - **geom_smooth** as opposed to **geom_histogram**.
- The attribute **alpha** can be used to control transparency.
 - **alpha** = 0 is completely transparent
 - **alpha** = 1 is completely opaque.

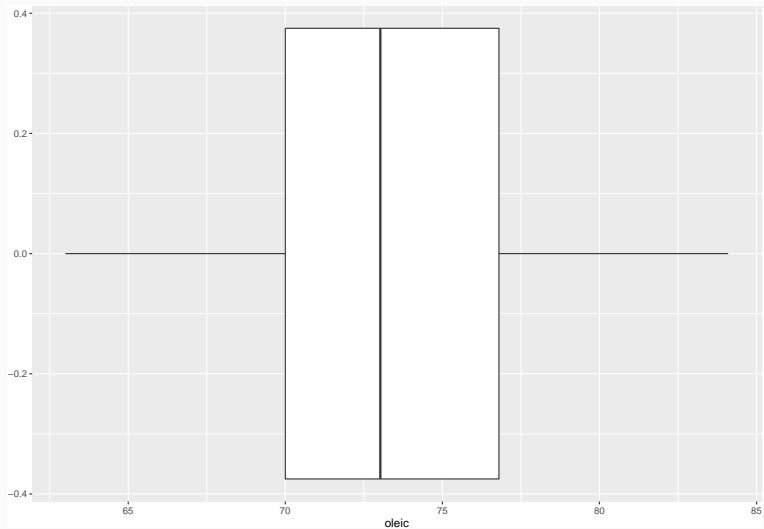
Boxplot i

- 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

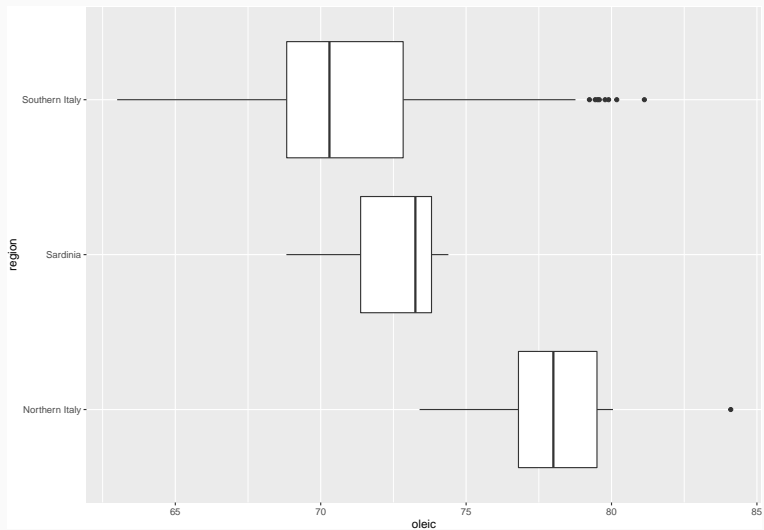
```
ggplot(olive, aes(x = oleic)) +  
  geom_boxplot()
```

Boxplot iii



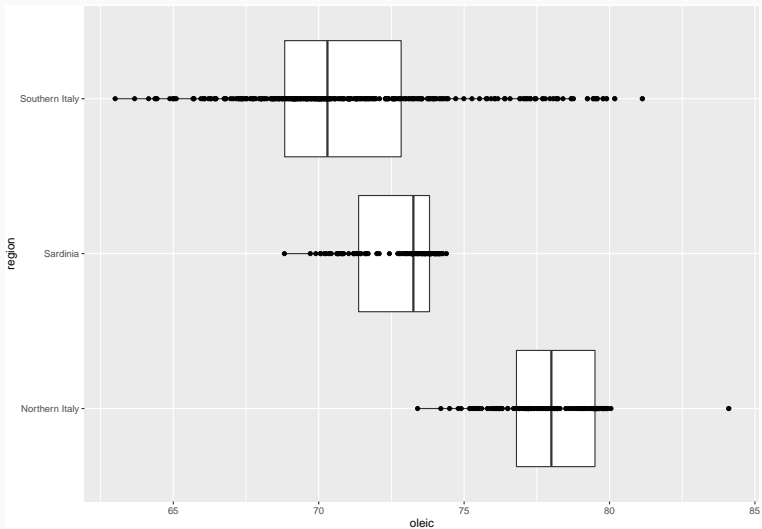
```
# Map region to y-axis  
ggplot(olive, aes(x = oleic,  
                  y = region)) +  
  geom_boxplot()
```

Boxplot v



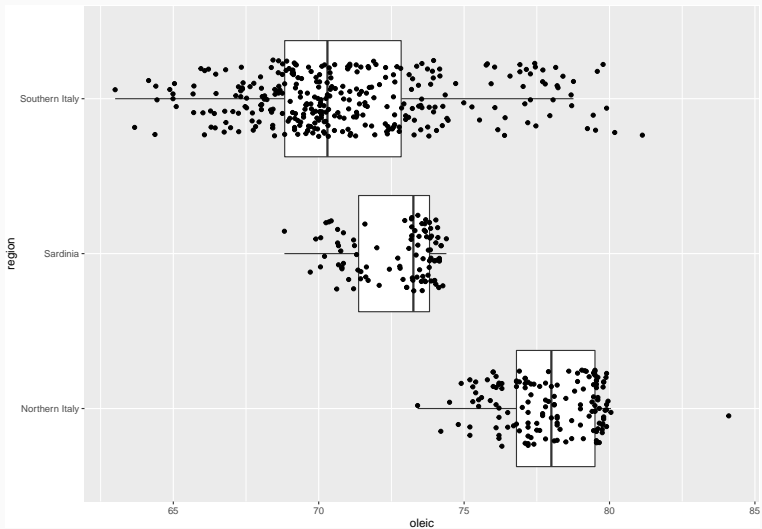
```
# Add all points on top of boxplots
ggplot(olive, aes(x = oleic,
                  y = region)) +
  geom_boxplot() +
  geom_point()
```

Boxplot vii



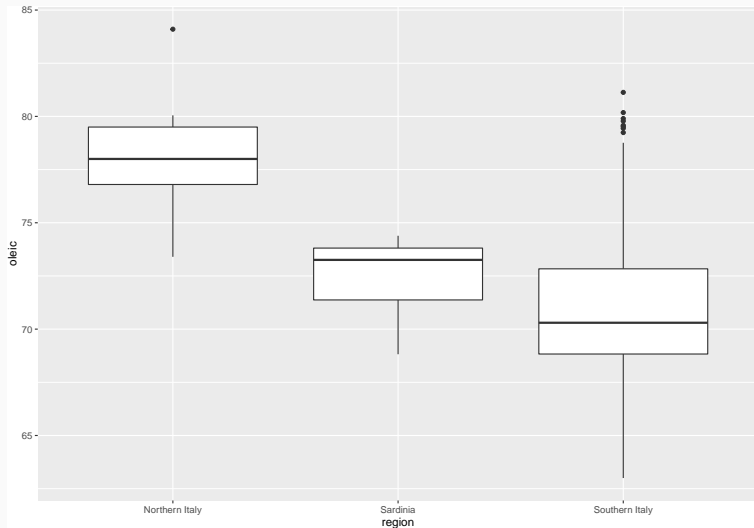
```
# Add vertical noise to the points to reduce overlap
# Note: need to remove outliers or you will get
#       duplicates
ggplot(olive, aes(x = oleic,
                  y = region)) +
  geom_boxplot(outlier.colour = NA) +
  geom_jitter(height = 0.25, width = 0)
```

Boxplot ix




```
# Flip boxplots by switching the axes  
ggplot(olive, aes(x = region,  
                  y = oleic)) +  
  geom_boxplot()
```

Boxplot xi



Boxplot—Summary

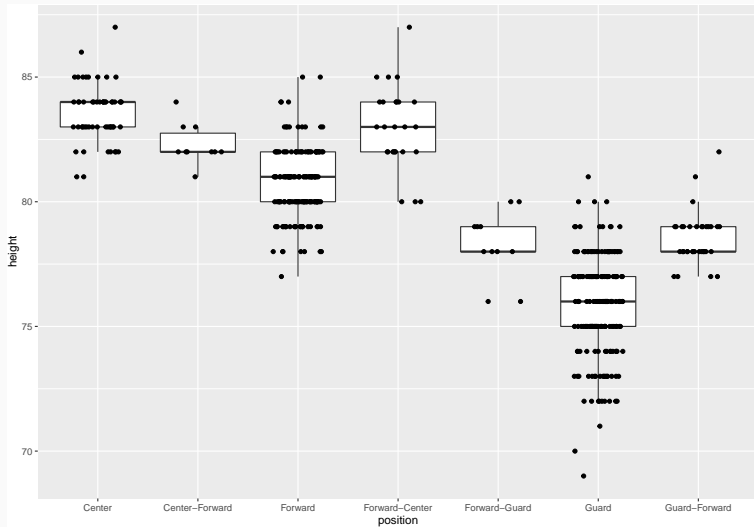
- **Boxplots** are a mixture between a data visualization and a summary statistics.
 - It is essentially a graphical depiction of the five-number summary.
- Widely different datasets can give rise to the same boxplot.
 - I recommend to overlay the actual data (potentially jittered).

Exercise

Using the dataset `nba_players_19` from the package `openintro`, compare the distribution of heights across all positions.

```
ggplot(nba_players_19, aes(x = position,  
                           y = height)) +  
  geom_boxplot(outlier.colour = NA) +  
  geom_jitter(height = 0, width = 0.25)
```

Solution ii



Single-variable visualization

- All three data visualizations above focused on a single continuous variable.
- But you can draw one such visualization for the same variable, but in different subgroups.
 - E.g. GPA for math, biology and psychology majors.
- In this way, they can all be used to investigate the relationships between *one continuous and one categorical variable*.

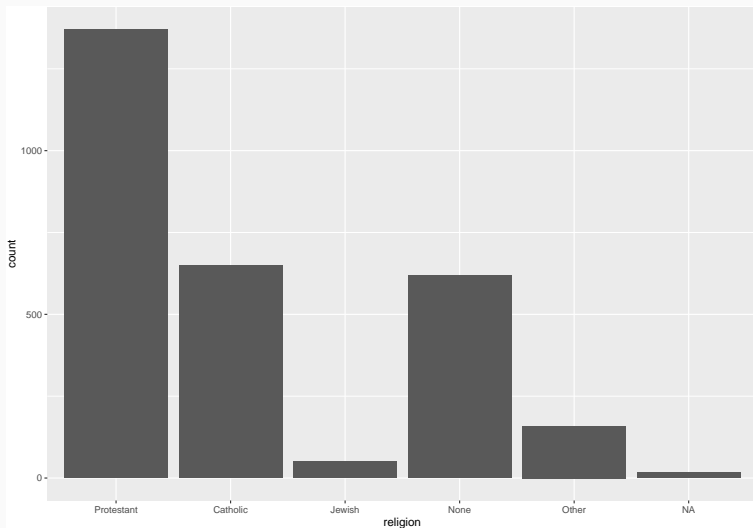
Bar plots i

- **Bar plots** are a very efficient way of displaying counts or percentages for different levels of a categorical variable.
 - Much, much better than pie charts
- When displaying summary statistics, they are also known as **dynamite plots**.
 - I **don't** recommend the use of dynamite plots.

```
library(socviz)
```

```
ggplot(gss_sm, aes(x = religion)) +  
  geom_bar()
```

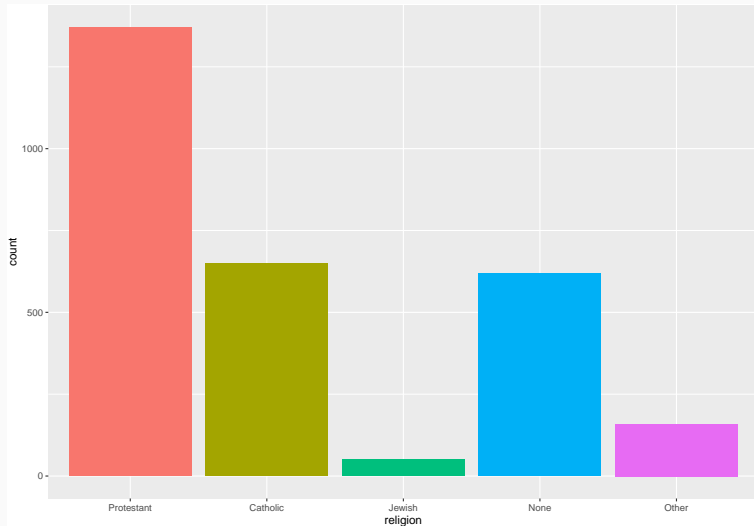

Bar plots ii



Bar plots iii

```
# Remove NA and add fill colour
gss_sm |>
  filter(!is.na(religion)) |>
  ggplot(aes(x = religion)) +
  geom_bar(aes(fill = religion),
           show.legend = FALSE)
```

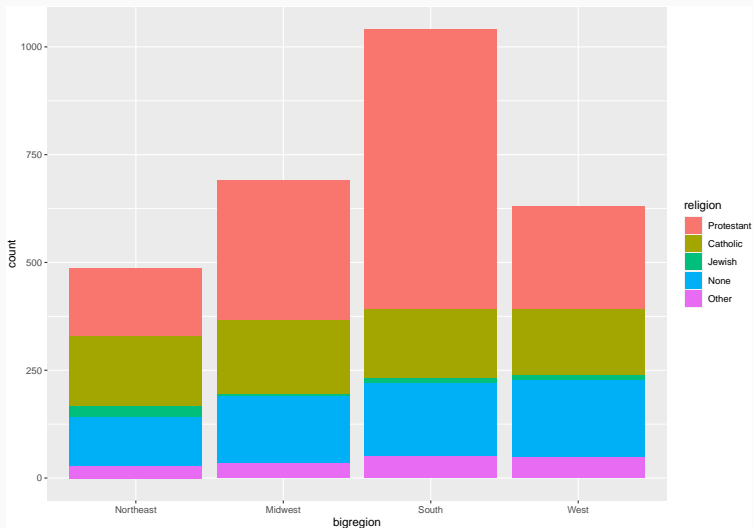
Bar plots iv



Bar plots v

```
# Across two categorical variables  
gss_sm |>  
  filter(!is.na(religion)) |>  
  ggplot(aes(x = bigregion)) +  
  geom_bar(aes(fill = religion))
```

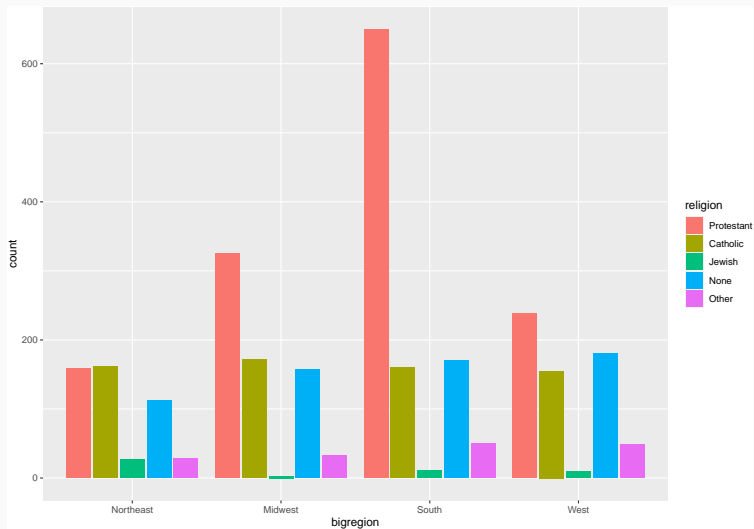
Bar plots vi



Bar plots vii

```
# Use dodge instead of stack
gss_sm |>
  filter(!is.na(religion)) |>
  ggplot(aes(x = bigregion)) +
  geom_bar(aes(fill = religion),
           position = "dodge2") # Adds some white space
```

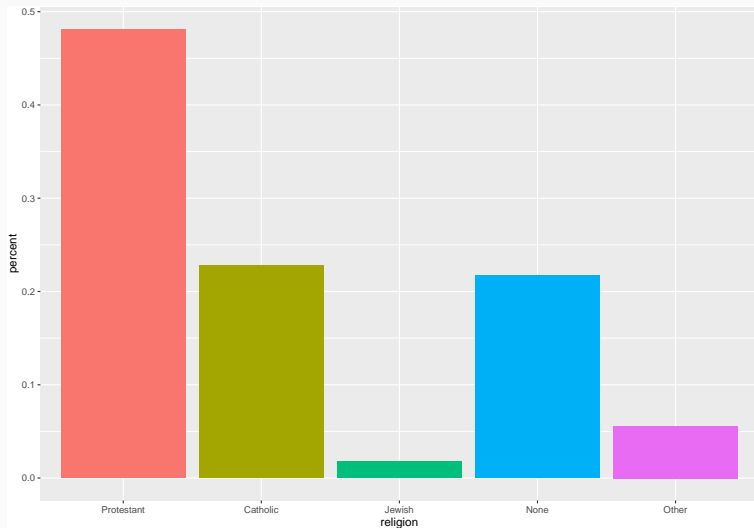
Bar plots viii



Bar plots ix

```
# For %, summarize first
gss_sm |>
  filter(!is.na(religion)) |>
  count(religion) |>
  mutate(percent = n/sum(n)) |>
  ggplot(aes(x = religion, y = percent)) +
  geom_col(aes(fill = religion),
           show.legend = FALSE)
```

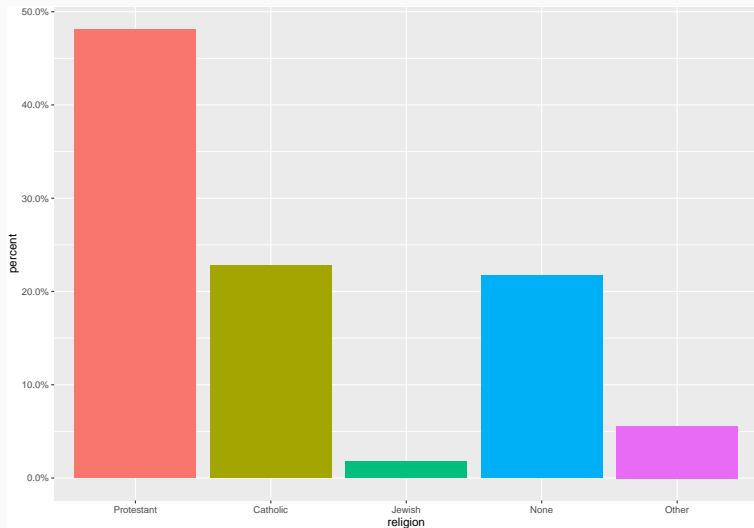

Bar plots x



Bar plots xi

```
# Turn into % by changing y scale
gss_sm |>
  filter(!is.na(religion)) |>
  count(religion) |>
  mutate(percent = n/sum(n)) |>
  ggplot(aes(x = religion, y = percent)) +
  geom_col(aes(fill = religion),
           show.legend = FALSE) +
  scale_y_continuous(labels = scales::percent)
```

Bar plots xii



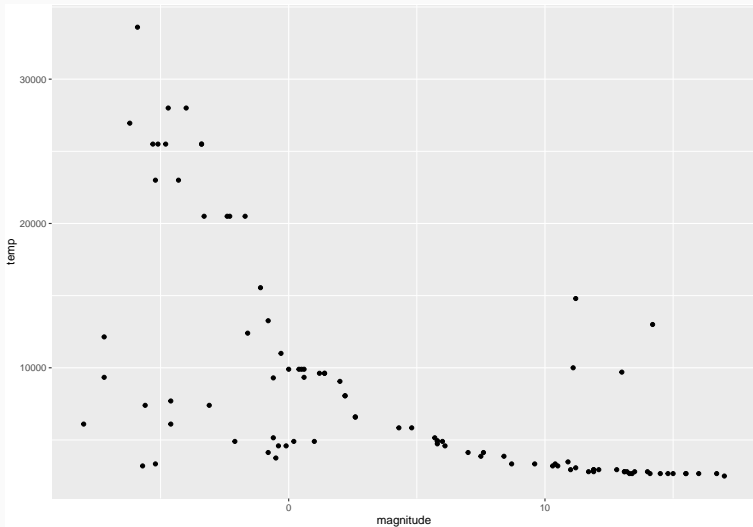
Bivariate plots

Scatter plot i

- The simplest way to represent the relationship between two continuous variables is a **scatter plot**.
 - Not really suitable with categorical variables.
- Technically still possible with three variables, but typically more difficult to read.

```
ggplot(stars, aes(x = magnitude,  
                  y = temp)) +  
  geom_point()
```

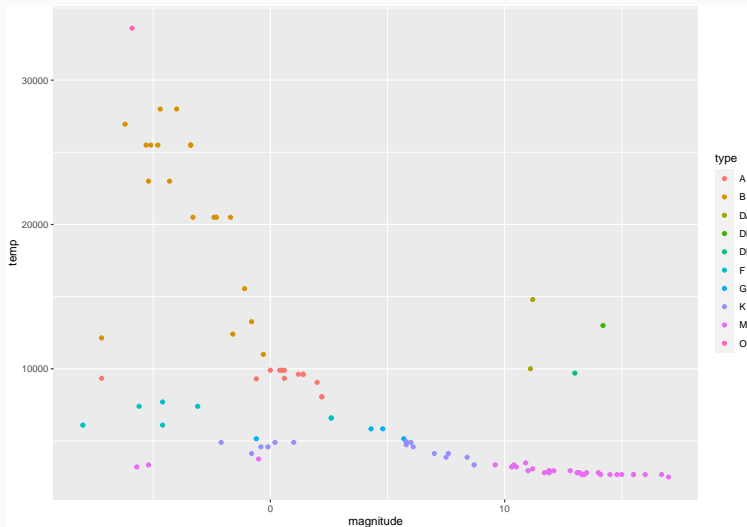
Scatter plot ii



Scatter plot iii

```
# Add colour for type of stars
ggplot(stars, aes(x = magnitude,
                  y = temp,
                  colour = type)) +
  geom_point()
```

Scatter plot iv



Exercise

Use the dataset `babies_crawl` from the package `openintro` to plot the average crawling age against the average outdoor temperature at 6 months.

Solution i

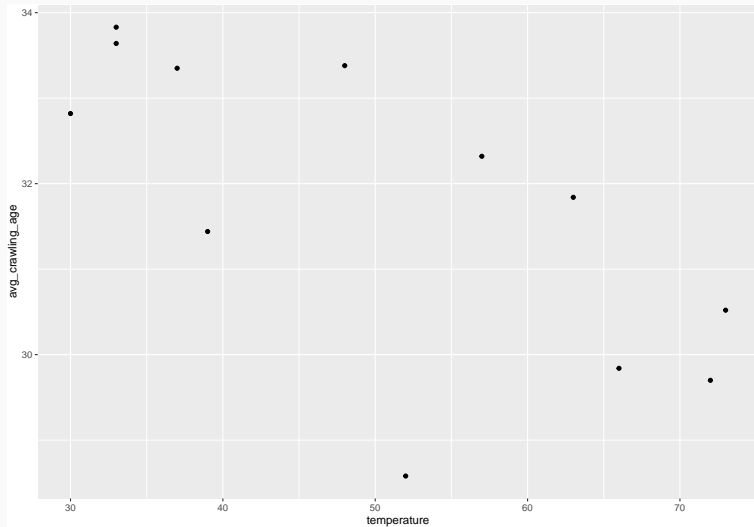
- First, we need to figure out the name of the variables we need to plot.
 - You can look at the help page `?babies_crawl`.
 - You can look at `str(babies_crawl)`.
- Our two variables are `temperature` and `avg_crawling_age`

Solution ii

```
library(tidyverse)
library(openintro)

ggplot(babies_crawl, aes(x = temperature,
                        y = avg_crawling_age)) +
  geom_point()
```

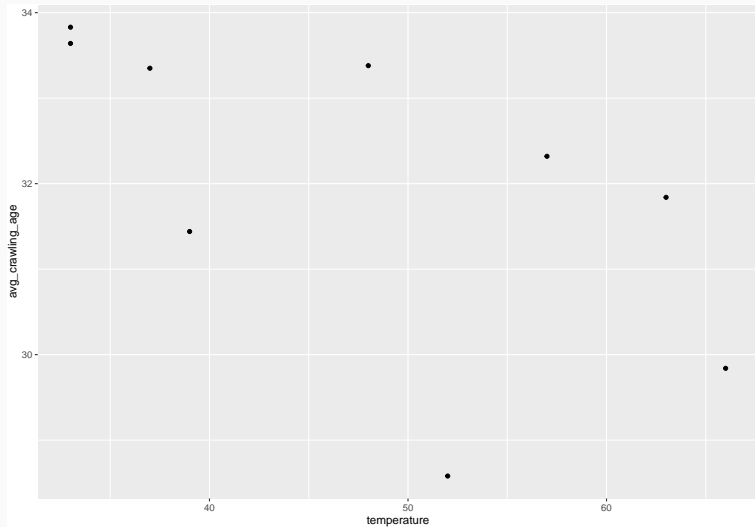
Solution iii



- What if we want to restrict the range of temperatures?

```
# First option
# Restrict the data before plotting
babies_crawl %>%
  filter(temperature > 30, temperature < 70) %>%
  ggplot(aes(x = temperature,
             y = avg_crawling_age)) +
  geom_point()
```

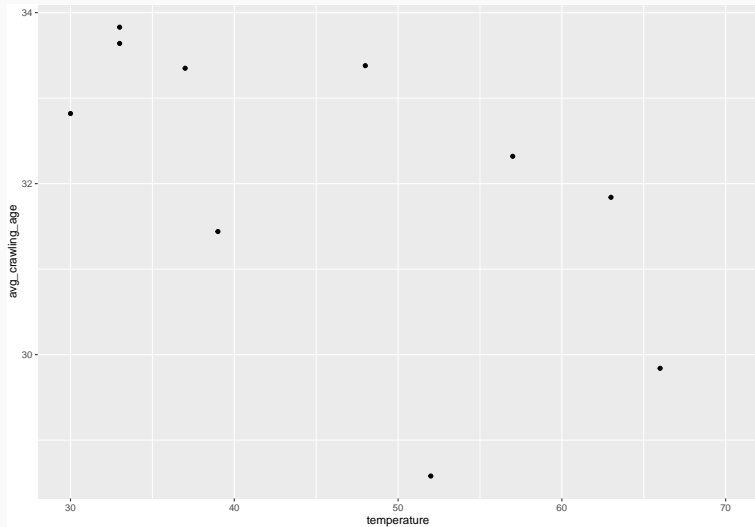
Solution v



```
# Second option
# xlim removes the points from the plot
ggplot(babies_crawl, aes(x = temperature,
                        y = avg_crawling_age)) +
  geom_point() +
  xlim(c(30, 70))

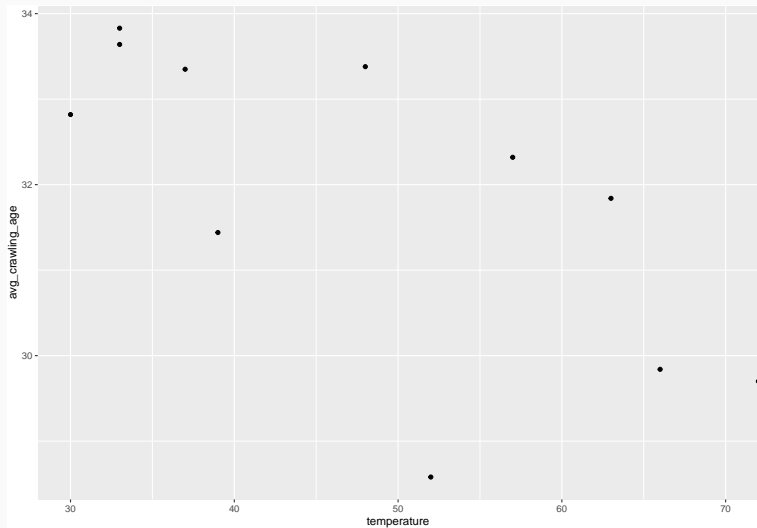
## Warning: Removed 2 rows containing missing values (ge
```

Solution vii




```
# Third option
# coord_cartesian zooms in/out
ggplot(babies_crawl, aes(x = temperature,
                        y = avg_crawling_age)) +
  geom_point() +
  coord_cartesian(xlim = c(30, 70))
```

Solution ix



Beyond two variables

Limitations

- Three-dimensional scatter plots are possible, but hard to interpret.
- 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 one at a time 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.

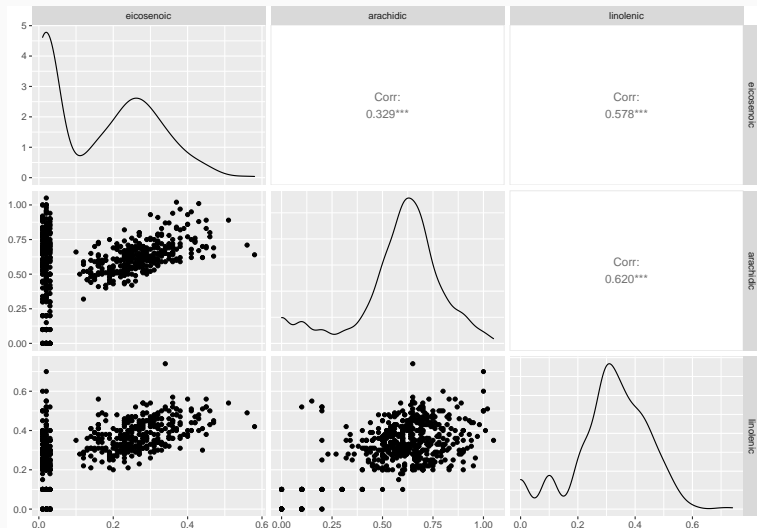
Pairs plot ii

```
library(GGally)

# Select three variables
olive_sub <- olive %>%
  select(eicosenoic, arachidic, linolenic)

ggpairs(olive_sub)
```

Pairs plot iii



- As we can see, **GGally** displays the following:
 - Scatter plots below the diagonal
 - Density plots on the diagonal
 - Pearson correlations above the diagonal
- These can all be changed—see the documentation for more information.