Using ggplot2

Data Analysis with R and Python

Deepayan Sarkar

ggplot2

- Traditional graphics and lattice share a common philosophy
  - Plotting functions are named after plot type
  - Example: `graphics::hist()` and `lattice::histogram()`

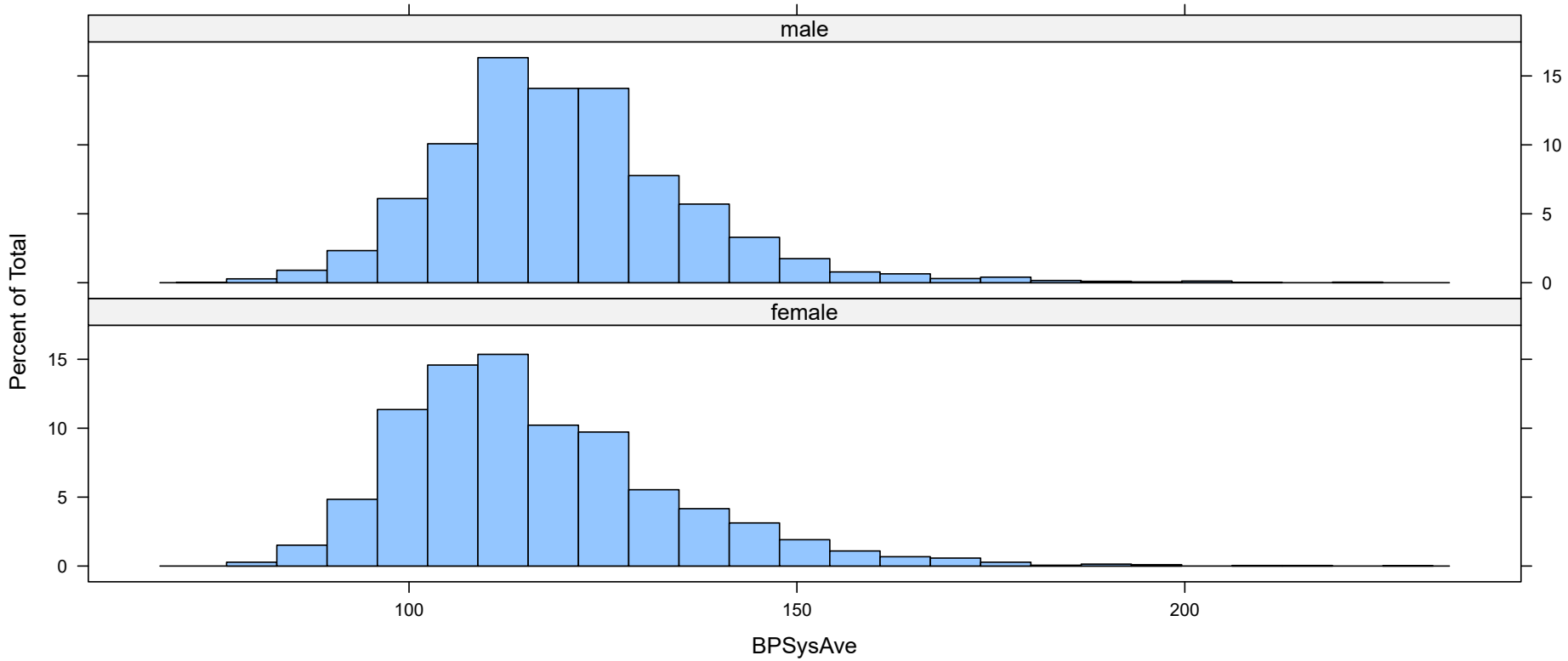
ggplot2

- Traditional graphics and lattice share a common philosophy
  - Plotting functions are named after plot type
  - Example: `graphics::hist()` and `lattice::histogram()`
- ggplot approaches data visualization in a different way

ggplot2

- Want to recreate the following histogram using ggplot2

```
data(NHANES, package = "NHANES")
lattice::histogram( ~ BPSysAve | Gender, data = NHANES,
                    layout = c(1, 2), nint = 25)
```



ggplot2

- The gg in ggplot stands for *grammar of graphics*
- Approach:
  - Define an abstract grammar for visualization
  - Consists of various predefined components
  - Plots are constructed by combining these components

ggplot2

- We start by specifying two of those components:
  - The dataset to be used
  - The faceting variable

```
library(package = "ggplot2")
p <- ggplot(data = NHANES) + facet_wrap(~ Gender, ncol = 1)
```

ggplot2

- Note that we have assigned the result to a variable

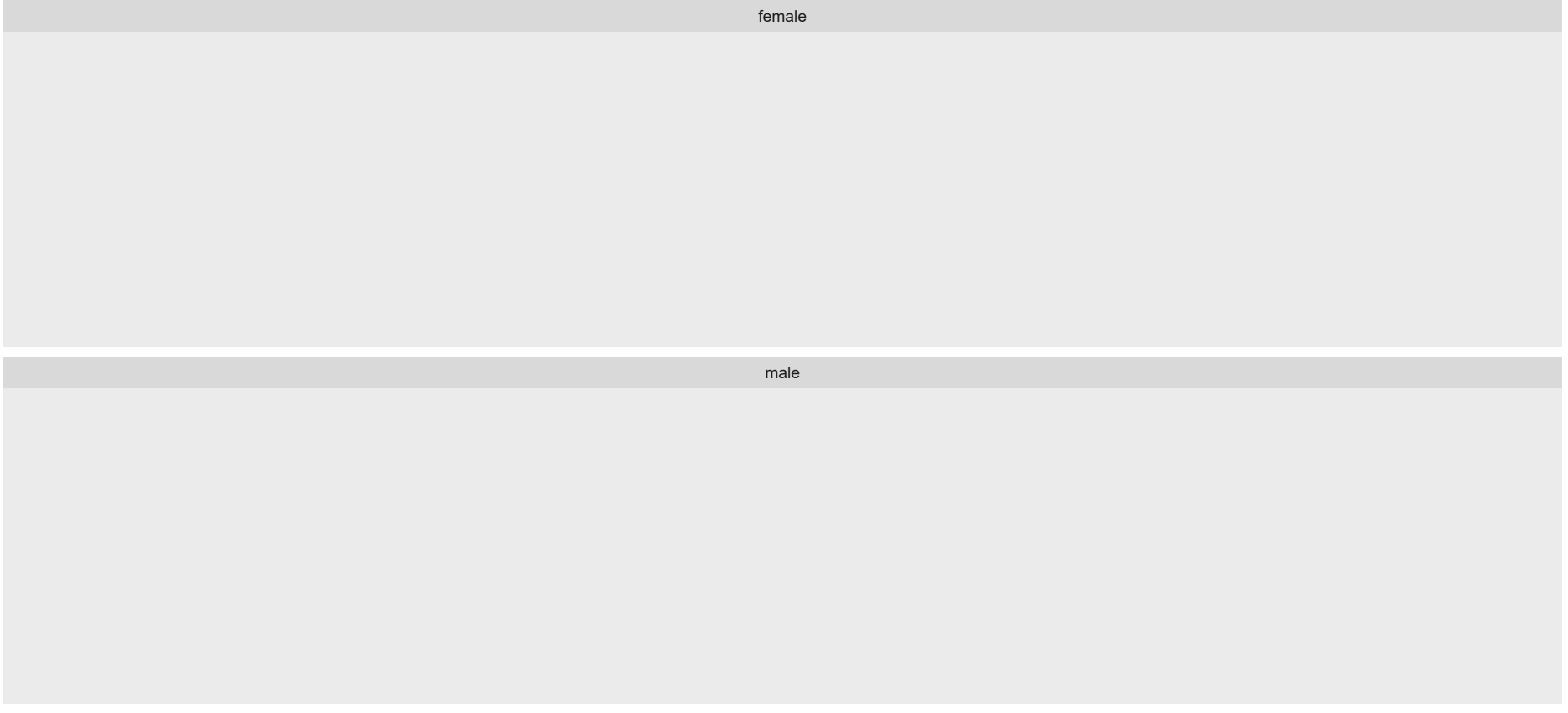
ggplot2

- Note that we have assigned the result to a variable
- This is an important feature of **lattice** and **ggplot**
  - Traditional graphics calls plot something as soon as they are called
  - This is not true for **lattice** and **ggplot**
  - Instead, they create objects that contain the information required for a plot

ggplot2

- Actual plot is created when they are 'printed' (blank because it doesn't have any geoms)

```
p
```

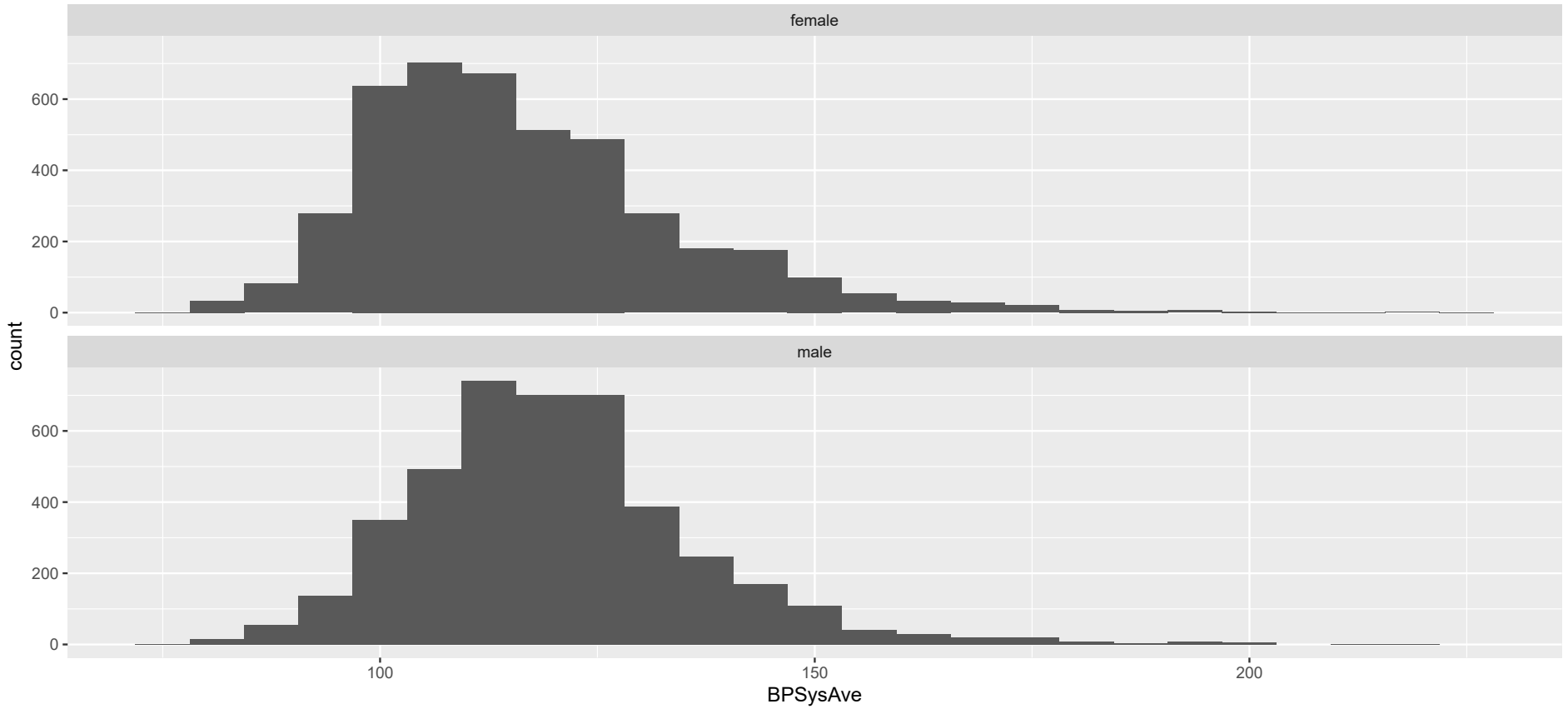


ggplot2

- Fortunately, there is a "histogram" geom we can use

```
p + geom_histogram(aes(x = BPSysAve), bins = 25)
```

```
Warning: Removed 1449 rows containing non-finite values (`stat_bin()`).
```



Stats

- This approach hides an intermediate step
- We map average systolic blood pressure to the x-coordinate (aesthetic mapping)
- However, the y-coordinate mapping is not explicitly mapped

Stats

- This approach hides an intermediate step
- We map average systolic blood pressure to the x-coordinate (aesthetic mapping)
- However, the y-coordinate mapping is not explicitly mapped
- What is actually plotted here are summary statistics (bin counts)
- Even the x-coordinate is no longer the raw data, but only the bin locations

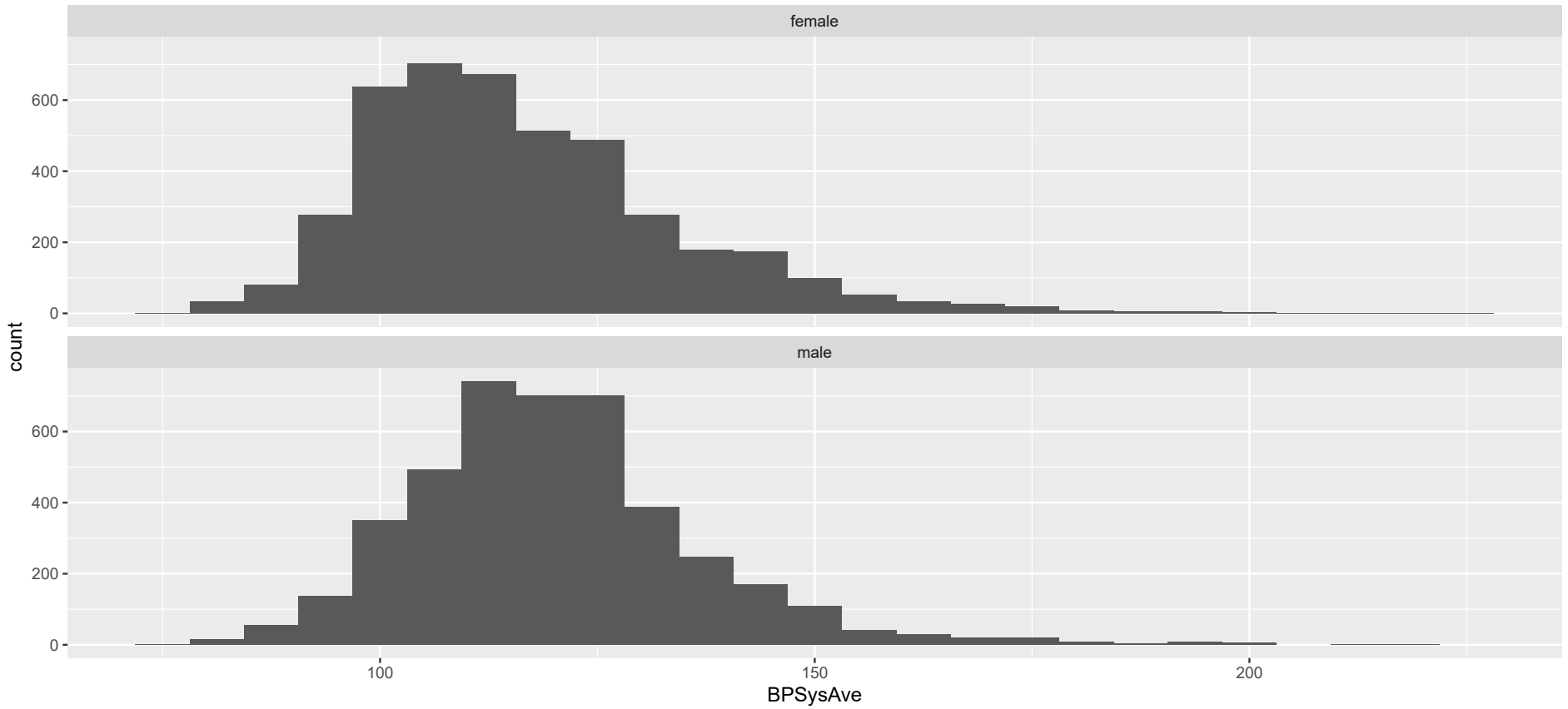
Stats

- This approach hides an intermediate step
- We map average systolic blood pressure to the x-coordinate (aesthetic mapping)
- However, the y-coordinate mapping is not explicitly mapped
- What is actually plotted here are summary statistics (bin counts)
- Even the x-coordinate is no longer the raw data, but only the bin locations
- So essentially raw data has been summarized to create some new data (which is plotted)
- Such statistical summaries are called **"stats"** in ggplot2 terminology

Stats

- We can make this operation explicit as follows

```
p + stat_bin(aes(x = BPSysAve), bins = 25, na.rm = TRUE)
```



Stats

- This 'stat' is applied by default in `geom_histogram()`

```
str(geom_histogram)
```

```
function (mapping = NULL, data = NULL, stat = "bin", position = "stack", ..., binwidth = NULL,
    bins = NULL, na.rm = FALSE, orientation = NA, show.legend = NA, inherit.aes = TRUE)
```

Stats

- This 'stat' is applied by default in `geom_histogram()`

```
str(geom_histogram)
```

```
function (mapping = NULL, data = NULL, stat = "bin", position = "stack", ..., binwidth = NULL,
    bins = NULL, na.rm = FALSE, orientation = NA, show.legend = NA, inherit.aes = TRUE)
```

- Similarly, `stat_bin()` has a default 'geom'

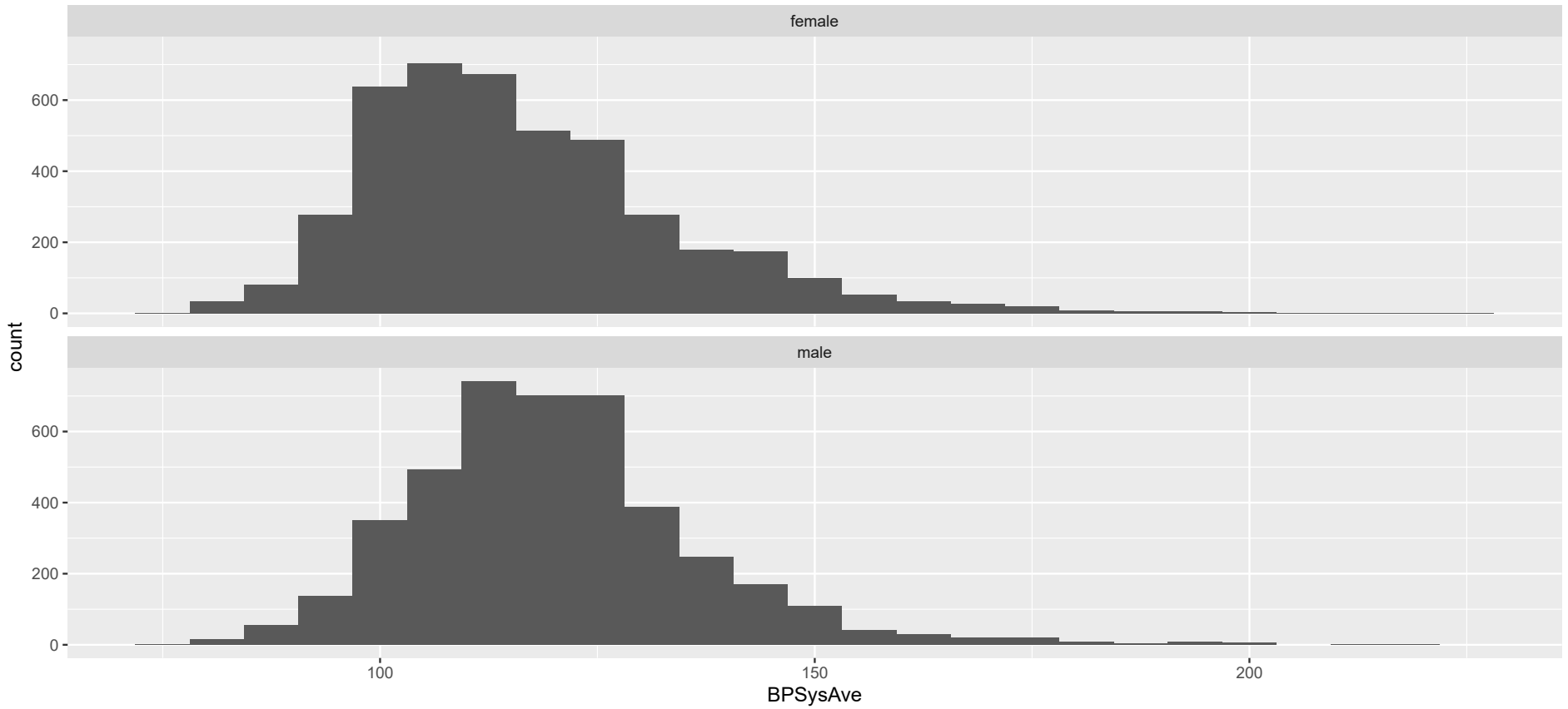
```
str(stat_bin)
```

```
function (mapping = NULL, data = NULL, geom = "bar", position = "stack", ..., binwidth = NULL,
    bins = NULL, center = NULL, boundary = NULL, breaks = NULL, closed = c("right",
        "left"), pad = FALSE, na.rm = FALSE, orientation = NA, show.legend = NA,
    inherit.aes = TRUE)
```

Stats

- Thus, the last plot we created is actually

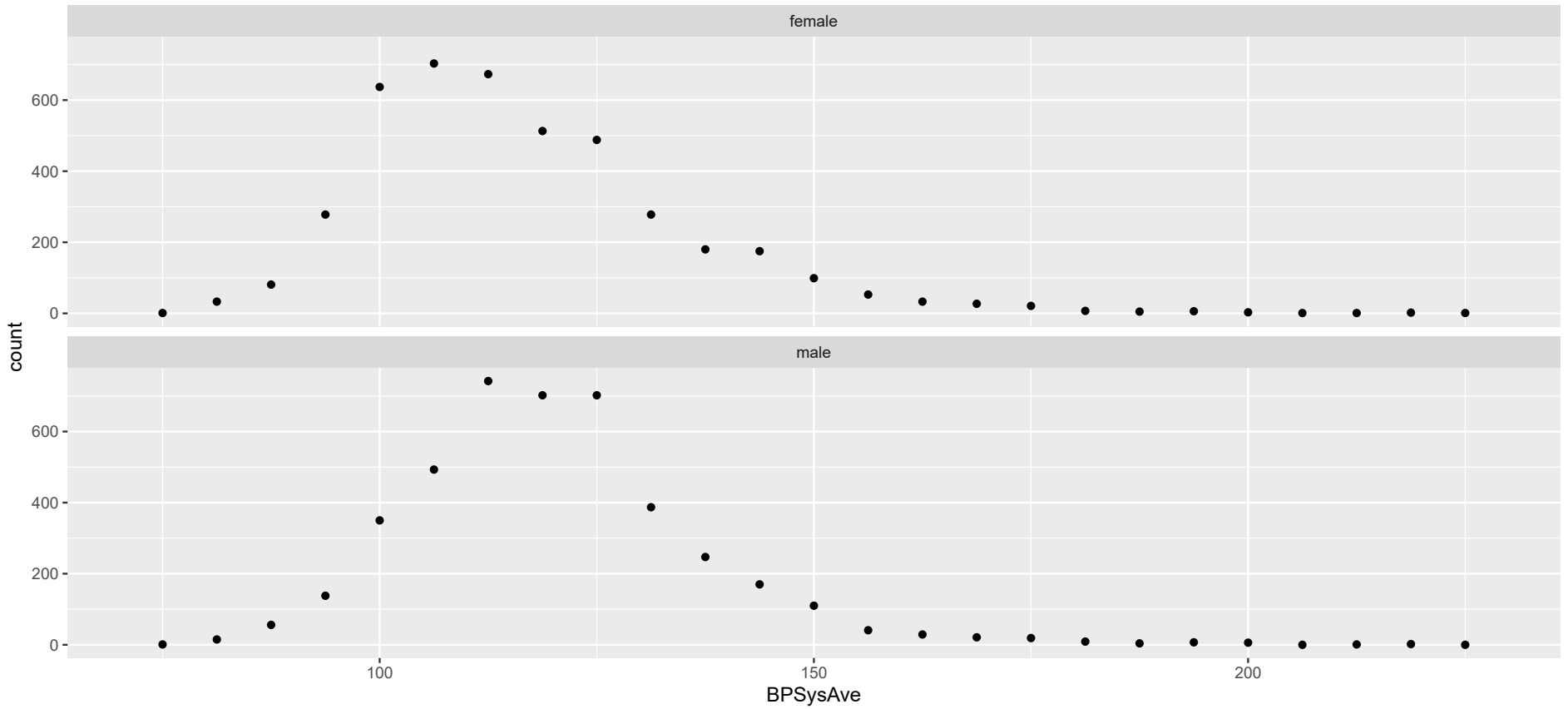
```
p + stat_bin(aes(x = BPSysAve), bins = 25, geom = "bar", na.rm = TRUE)
```



Stats

- We can change the geom to get a different visualization of the same summary

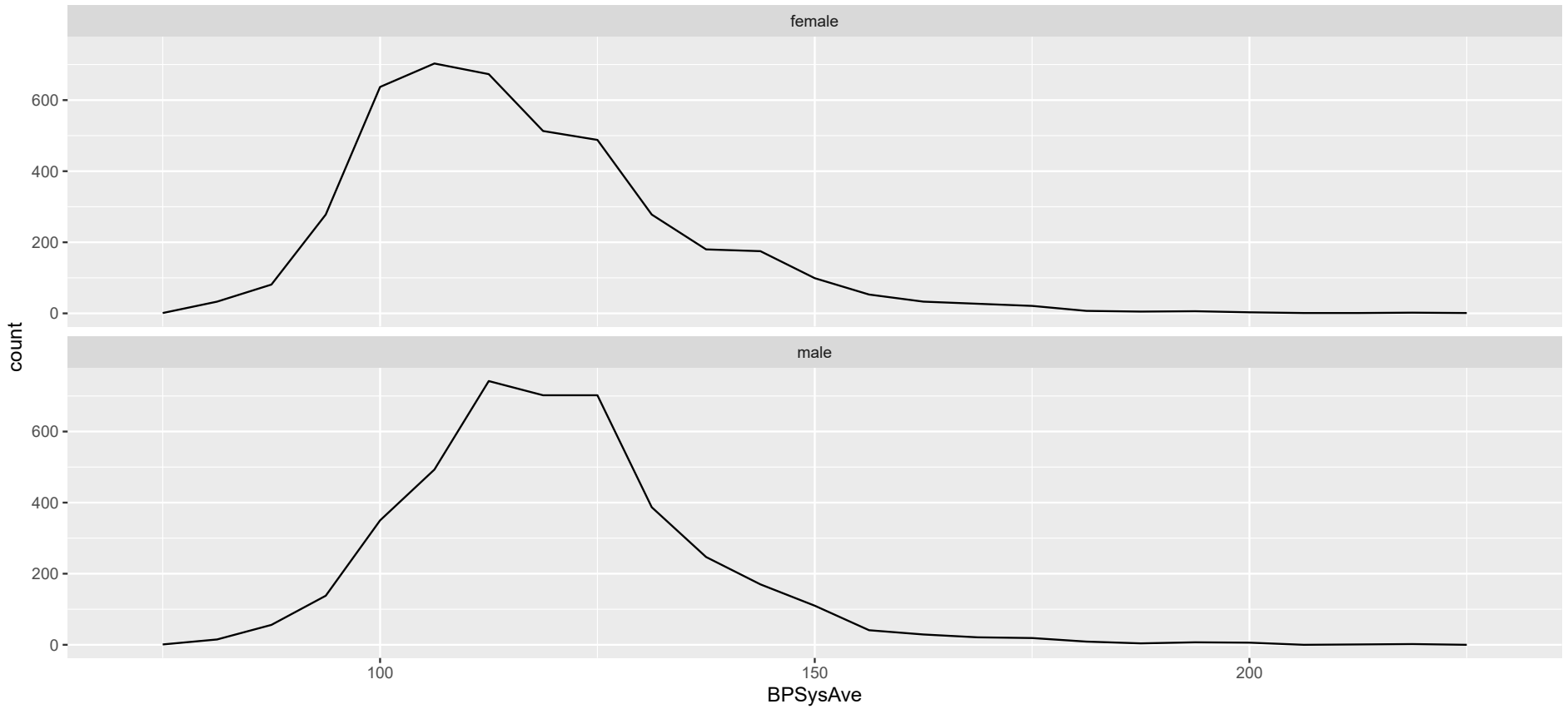
```
p + stat_bin(aes(x = BPSysAve), bins = 25, geom = "point", na.rm = TRUE)
```



Stats

- A more useful choice of geom is "line" (result is a reasonable substitute for histograms)

```
p + stat_bin(aes(x = BPSysAve), bins = 25, geom = "line", na.rm = TRUE)
```



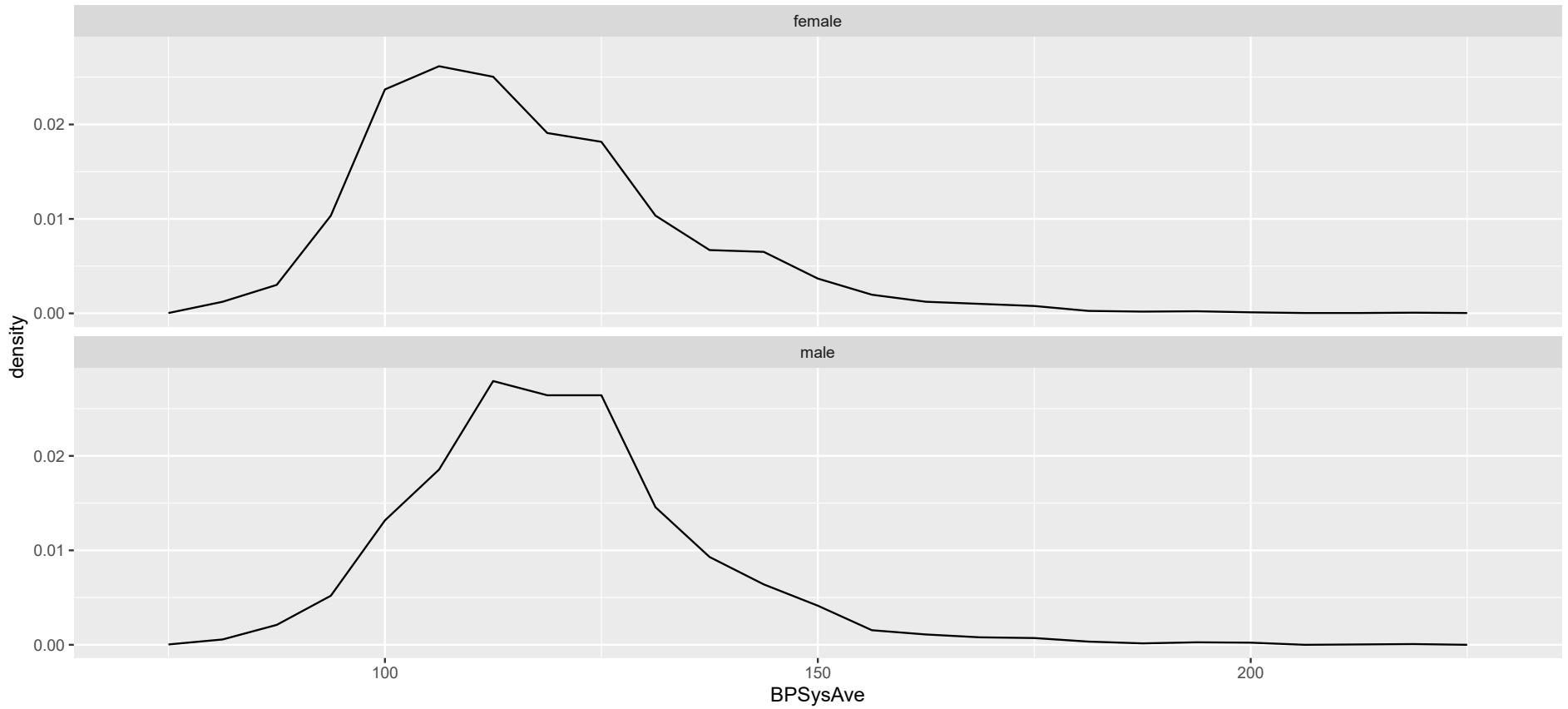
Stats

- These plots show bin counts
- As we saw earlier, histograms can also be viewed as *density* estimators
- Read help page of `stat_bin()`: it also computes estimated density

Stats

- For these, y-axis mapping must be `density`, not `count`

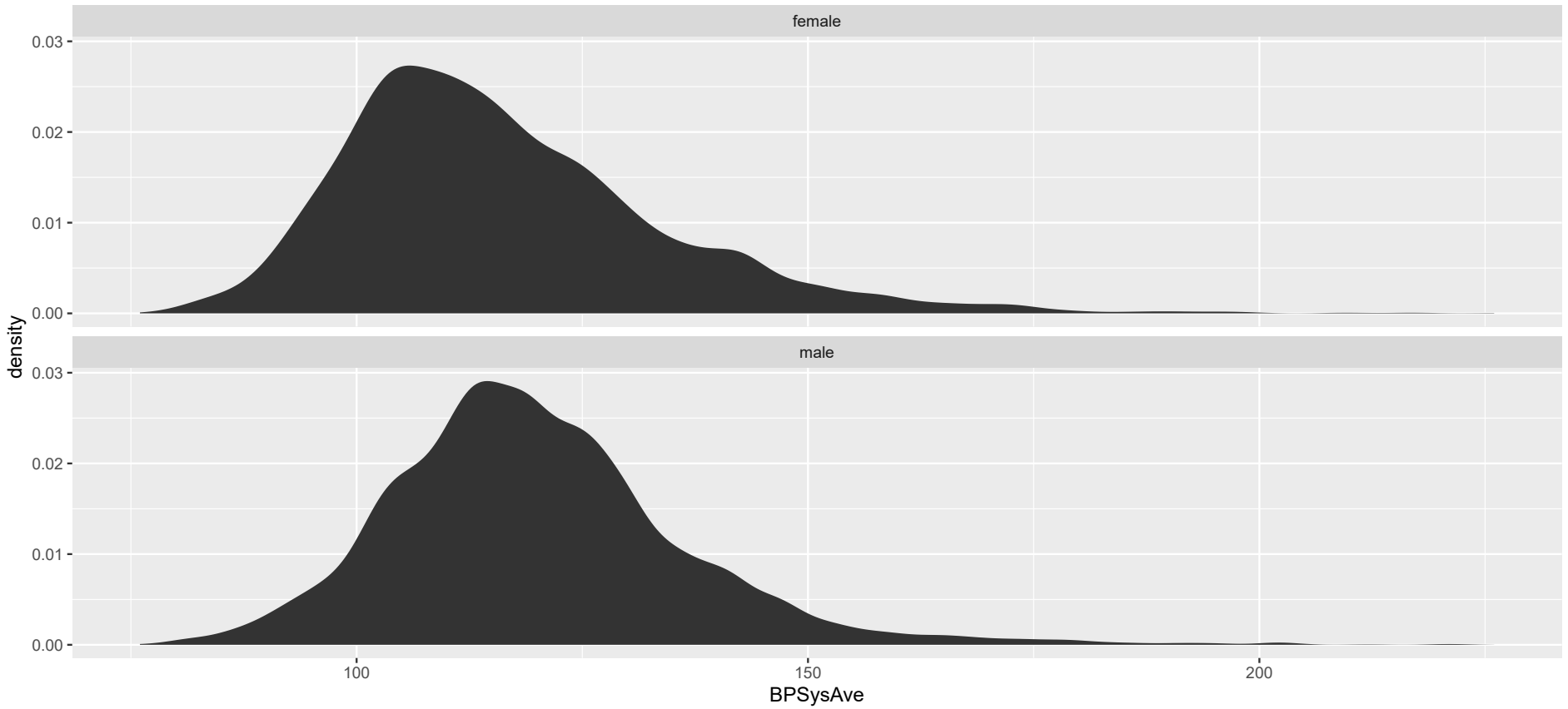
```
p + stat_bin(aes(x = BPSysAve, y = after_stat(density)), bins = 25, geom = "line", na.rm = TRUE)
```



Stats

- Alternative method: average shifted histogram / kernel density estimation

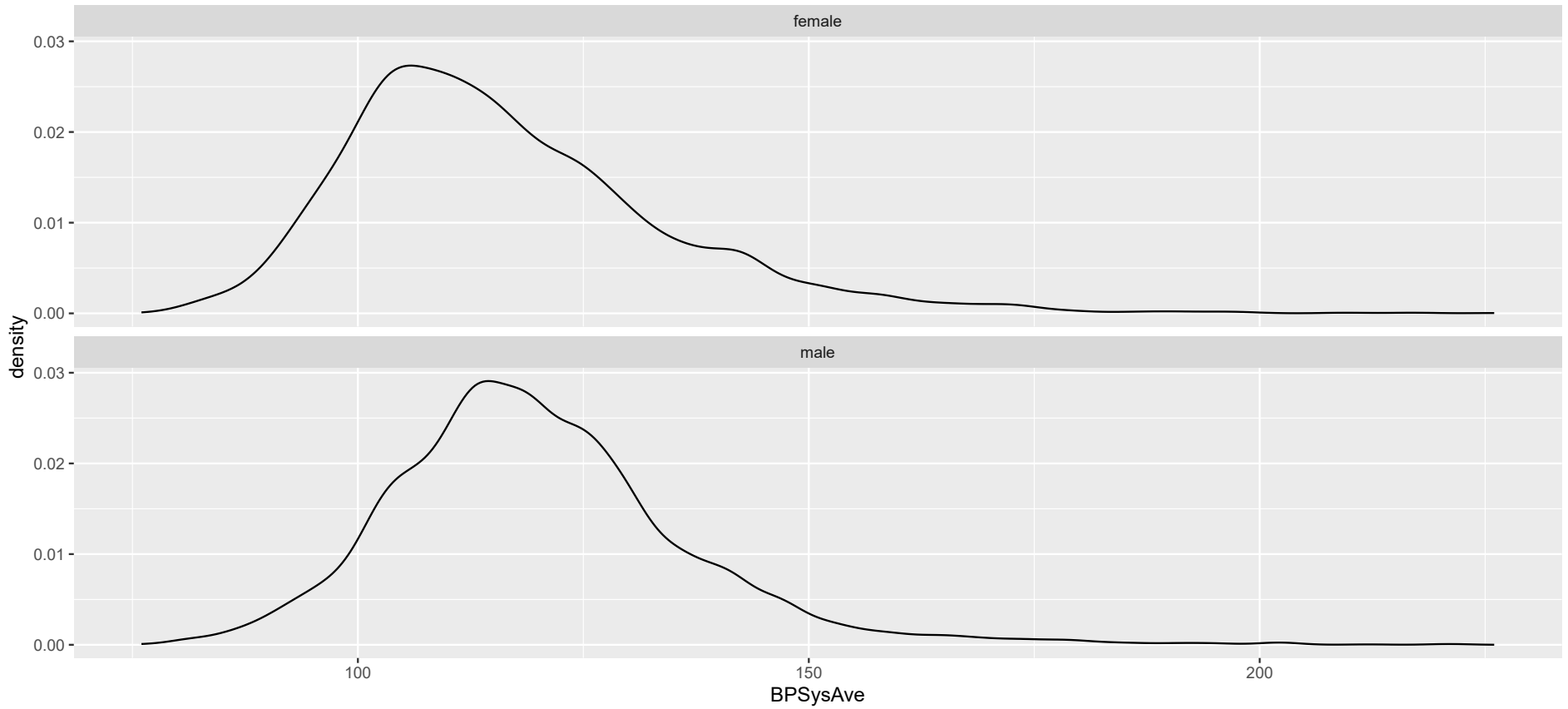
```
p + stat_density(aes(x = BPSysAve), na.rm = TRUE)
```



Stats

- Again, replacing the default geom ("area") by "line" gives more familiar density plot

```
p + stat_density(aes(x = BPSysAve), geom = "line", na.rm = TRUE)
```



Aesthetic mapping

- Important feature of ggplot2: *aesthetic mapping*
  - Aesthetic attributes of any geom can be mapped to specific variables
  - This includes, color, fill, size, shape, etc.
  - Details depend on specific geom

Aesthetic mapping

- Important feature of ggplot2: *aesthetic mapping*
  - Aesthetic attributes of any geom can be mapped to specific variables
  - This includes, color, fill, size, shape, etc.
  - Details depend on specific geom

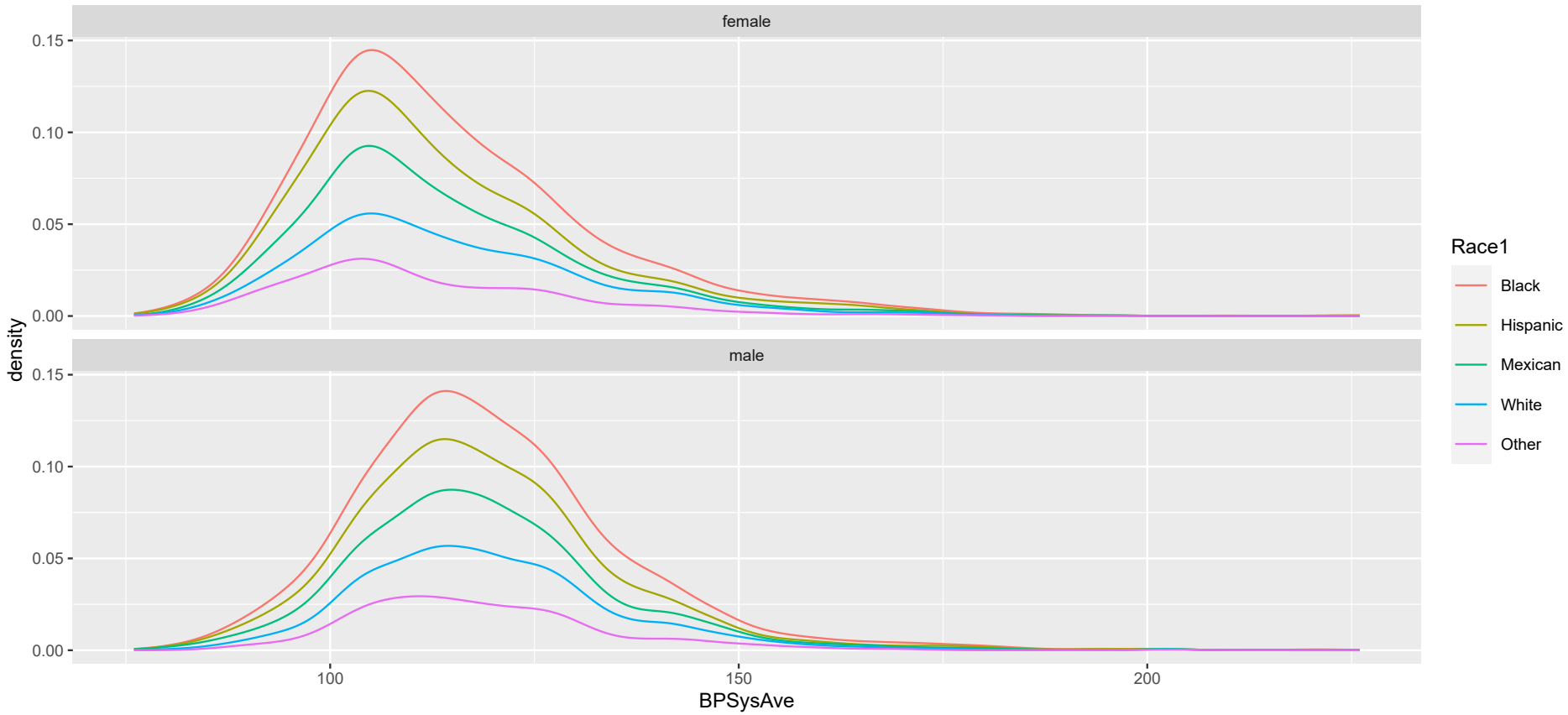
```
str(geom_line)
```

```
function (mapping = NULL, data = NULL, stat = "identity", position = "identity",
    na.rm = FALSE, orientation = NA, show.legend = NA, inherit.aes = TRUE, ...)
```

Aesthetic mapping

- Example: map `Race1` (categorical variable) to line color

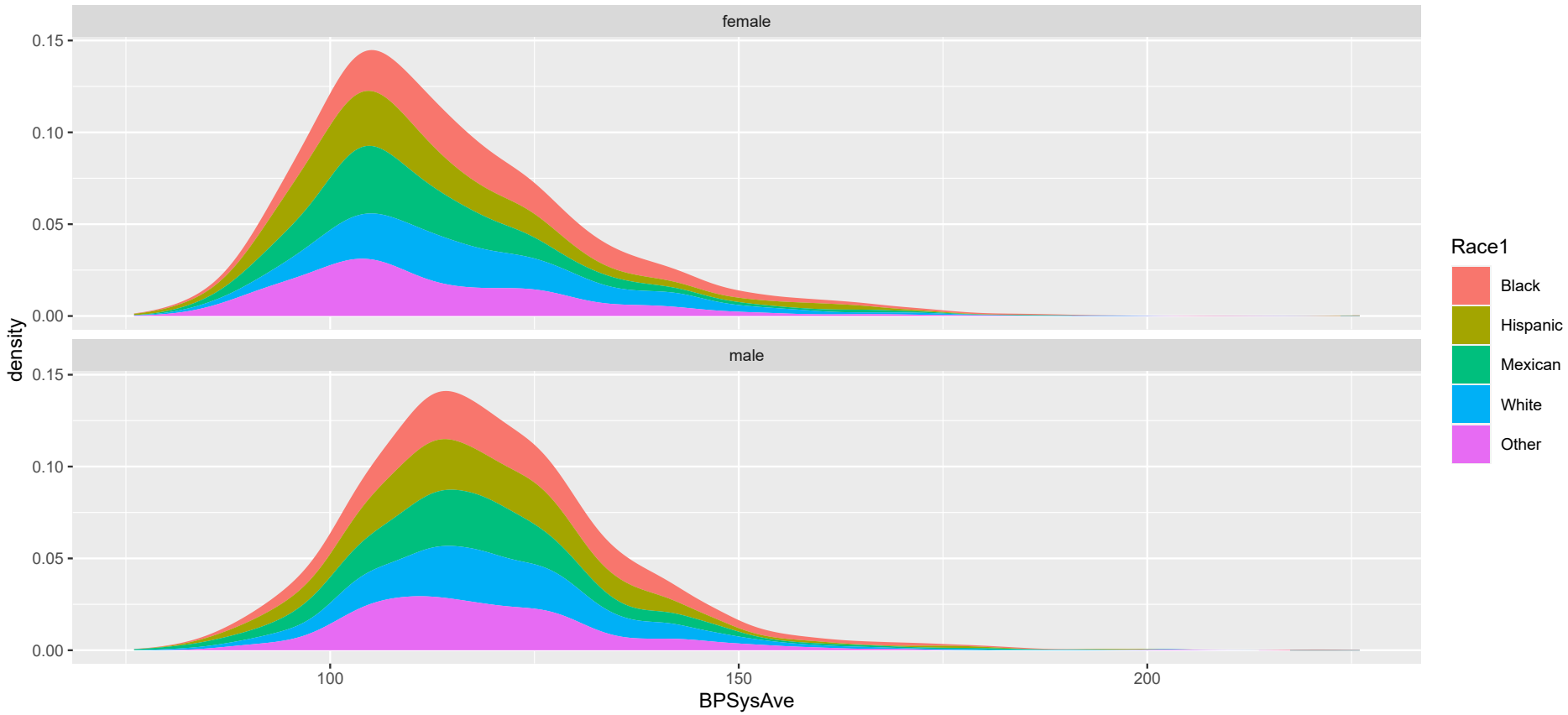
```
p + stat_density(aes(x = BPSysAve, color = Race1), geom = "line", na.rm = TRUE)
```



Aesthetic mapping

- Example: Default `position = "stack"` is useful for default geom "area"

```
p + stat_density(aes(x = BPSysAve, fill = Race1), geom = "area", na.rm = TRUE)
```



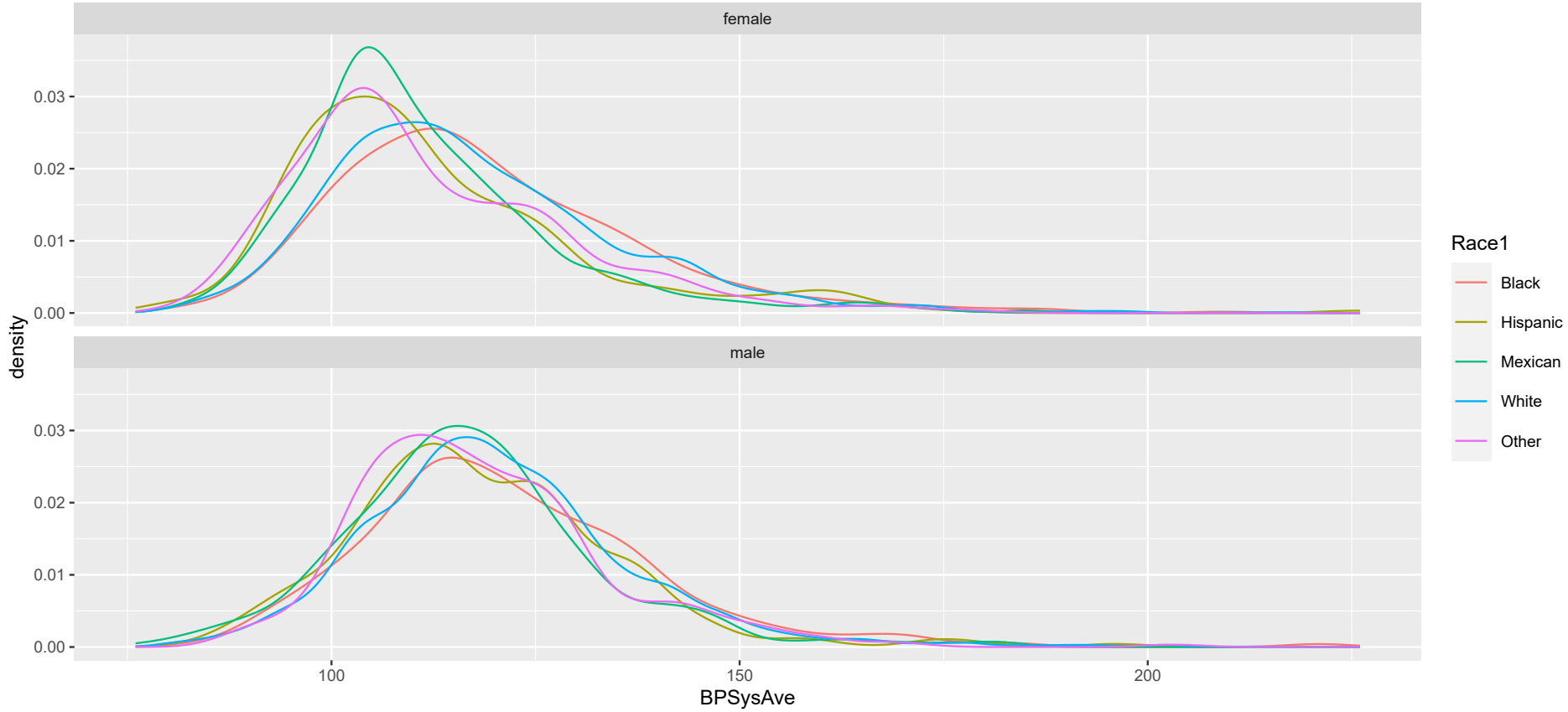
Aesthetic mapping

```
str(stat_density)
```

```
function (mapping = NULL, data = NULL, geom = "area", position = "stack", ..., bw = "nrd0",
    adjust = 1, kernel = "gaussian", n = 512, trim = FALSE, na.rm = FALSE, bounds = c(-Inf,
        Inf), orientation = NA, show.legend = NA, inherit.aes = TRUE)
```

Aesthetic mapping

```
p + stat_density(aes(x = BPSysAve, color = Race1),
                 geom = "line", position = "identity", na.rm = TRUE)
```



Aesthetic mapping

- The idea of aesthetic mapping of variables to visual attributes is very powerful
- We will demonstrate this with the gapminder data

```
library(gapminder)
str(gapminder)
```

```
tibble [1,704 × 6] (S3: tbl_df/tbl/data.frame)
 $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ year     : int [1:1704] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
 $ lifeExp  : num [1:1704] 28.8 30.3 32 34 36.1 ...
 $ pop      : int [1:1704] 8425333 9240934 10267083 11537966 13079460 14880372 12881816 13867957 16317921
22227415 ...
 $ gdpPercap: num [1:1704] 779 821 853 836 740 ...
```

Aesthetic mapping

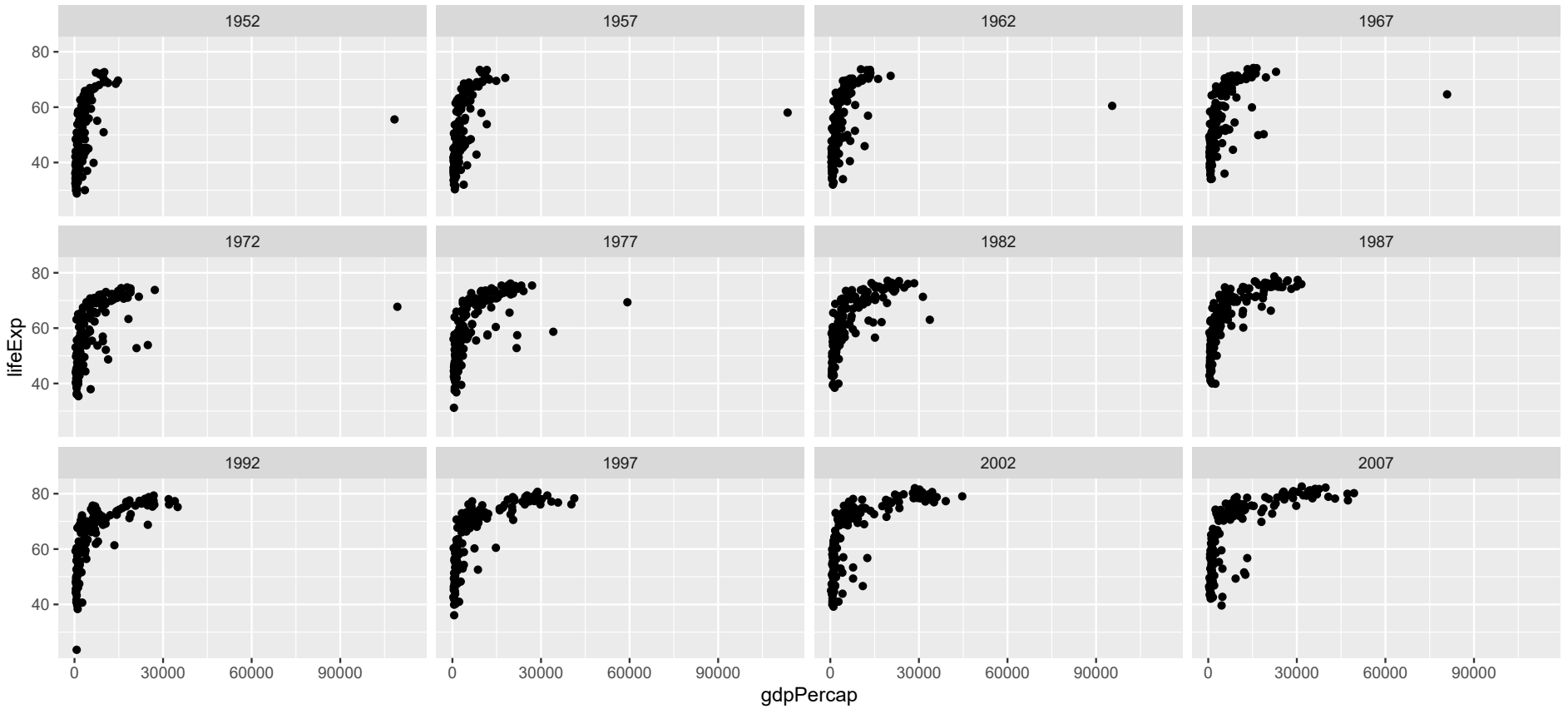
- Create a blank ggplot object with data and a faceting variable (year)

```
pgm <- ggplot(data = gapminder) + facet_wrap(~ year)
```

Aesthetic mapping

- Scatter plot to study relation between `lifeExp` and `gdpPercap`

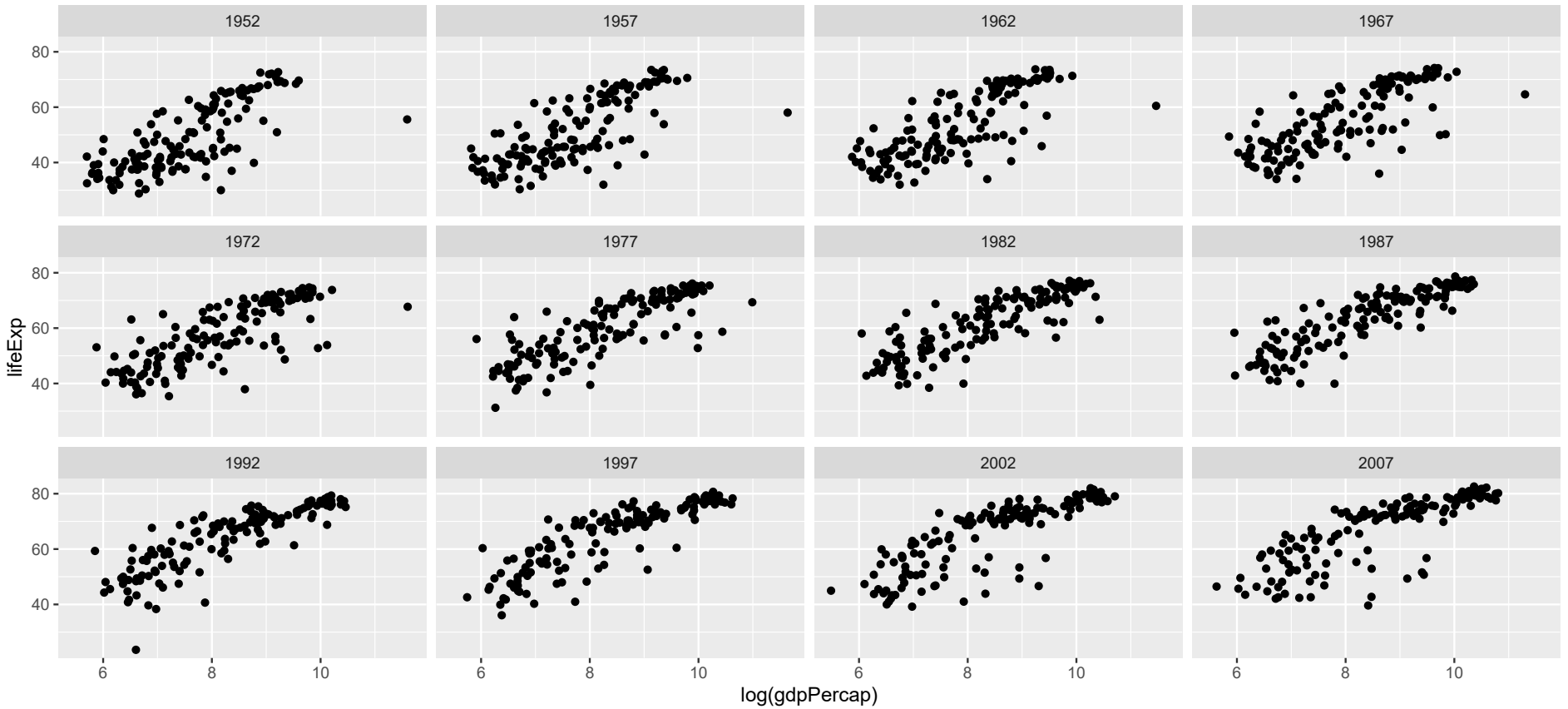
```
pgm + geom_point(aes(x = gdpPercap, y = lifeExp))
```



Aesthetic mapping

- `gdpPercap` is highly skewed, so take logarithm

```
pgm + geom_point(aes(x = log(gdpPercap), y = lifeExp))
```



Aesthetic mapping

- Other attributes: map continent to color

```
pgm + geom_point(aes(x = log(gdpPercap), y = lifeExp, color = continent))
```

Aesthetic mapping

- Another continuous variable is population: map to *size* of points (bubble chart)

```
pgm + geom_point(aes(x = log(gdpPercap), y = lifeExp, color = continent, size = pop))
```