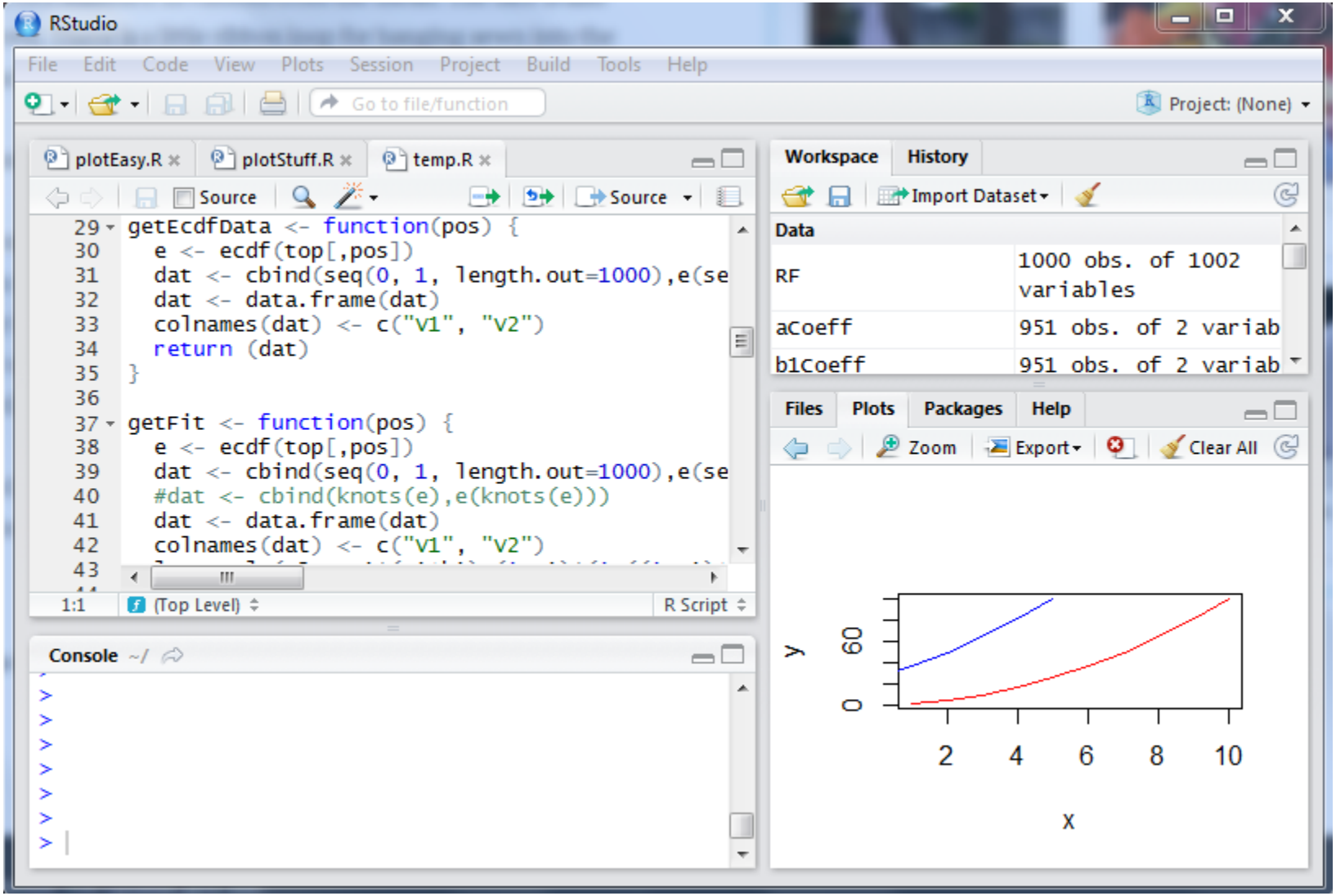
# День 1

# Зачем нужен R?

- Быстрая статистическая обработка данных
- Построение красивых графиков
- Бесплатный, удобный, быстрый для изучения язык

# Среда разработки RStudio: http://www.r-studio.com/



# Помощь

- Форумы: Stackoverflow, R mailing list, etc.
- Документация (http://www.r-project.org, help(...))
- Наши лекции и тьюториалы

# R – векторизованный язык

- Основной тип данных – вектор (упорядоченный набор чисел)
- Идея – работать с набором данных как с одним числом (параллельно обрабатывать все значения набора)
- Это позволяет обходиться (в ряде случаев) без циклов

# Вектор

```
> x<-1:5 ; y<-6:10
> x
[1] 1 2 3 4 5
> y
[1]  6  7  8  9 10
> x+y
[1]  7  9 11 13 15
```

```
> x*2
[1]  2  4  6  8 10
> x>4
[1] FALSE FALSE FALSE
FALSE  TRUE
> y==7
[1] FALSE  TRUE FALSE
FALSE FALSE
> x*y
[1]  6 14 24 36 50
```

# Как можно создать вектор?

Оператор c()

```
> c(1, 2, 3)
[1] 1 2 3
```

Последовательности

```
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
> seq(from=1, to=8, by=2)
[1] 1 3 5 7
> seq(1, 10, 2)
[1] 1 3 5 7 9
```

# Как можно создать вектор?

Объединение нескольких векторов

```
> x<-c(1, 2, 3)
> x<-c(x, 1:3); x
[1] 1 2 3 1 2 3
```

Повторы

```
 > rep(0.5, 6)
[1] 0.5 0.5 0.5 0.5 0.5 0.5
```

Для целых чисел (работает быстрее)

```
> rep.int(1, 5)
[1] 1 1 1 1 1
```

# Как можно создать вектор?

**Распределение**

- Нормальное распределение:
- dnorm(x) – плотность распределения
- pnorm(q) – функция распределения
- qnorm(p) – квантильная функция

**Случайная генерация из распределения:**

```
> set.seed(100)
> rnorm(5)
[1]  1.1568405 -0.8248219  0.1428891 -0.4784408  0.7561443
```

**Равномерное**

runif(n, min=0, max=1)

> runif(5, 0, 1)

[1] 0.1972687 0.3090867 0.2865924 0.1409635 0.3441481

**Биномиальное**

rbinom(n, size, prob)

> rbinom(10, 100, 0.5)

[1] 54 47 55 50 47 45 52 45 58 52

**Пуассона**

rpois(n, lambda)

> rpois(10, 4)

[1] 2 3 2 4 10 3 2 3 5 6

# Срезы

```
> x<-c(1, 5, 7, 9, 15, 3)
> x[1]
[1] 1
> x[2:4]
[1] 5 7 9
> x[c(2, 5)]
[1]  5 15
```

```
> x[-1]
[1]  5  7  9 15  3
> x[-(1:3)]
[1]  9 15  3
> x[x>5]
[1]  7  9 15
> x[x>5 & x<10]
[1] 7 9
```

# Простейший статистический анализ

```
> x <- rnorm(100)
```

**Среднее**

```
> mean(x)
[1] -0.04029328
```

**Стандартное отклонение**

```
> sd(x)
[1] 1.037552
```

# Простейший статистический анализ

**Минимальное и максимальное значения**

```
> min(x)
[1] -2.605444
> max(x)
[1] 2.51254
```

**Медиана**

```
> median(x)
[1] -0.1039548
```

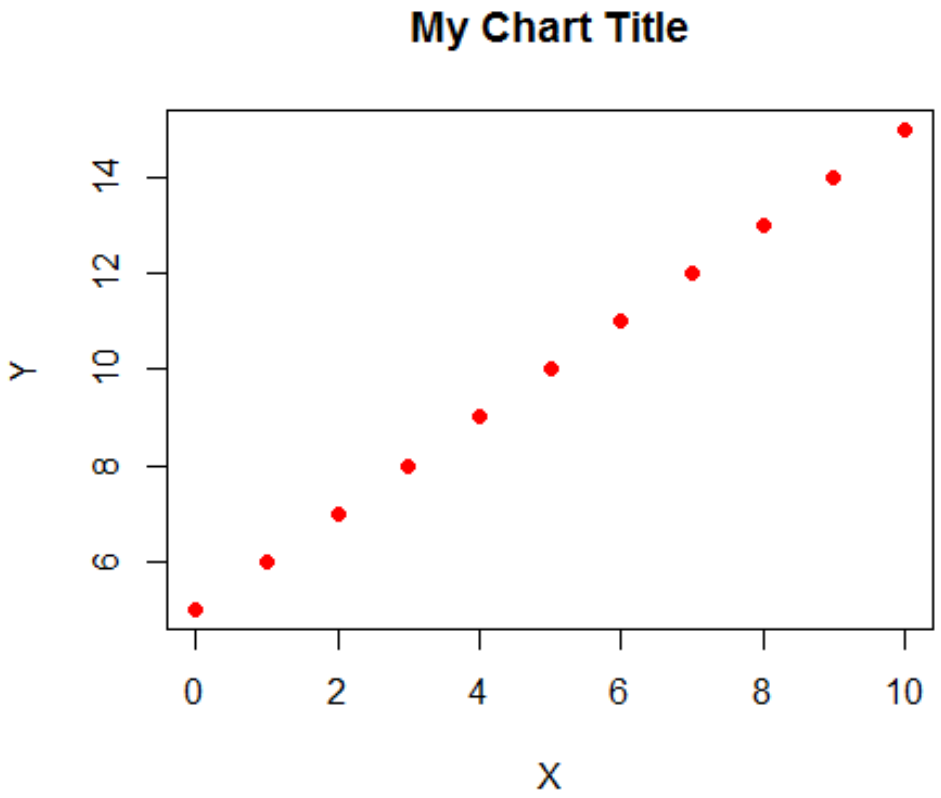
**Квантили**

```
> quantile(x)
       0%        25%        50%        75%
100%
-2.6054443 -0.6321819 -0.1039548
0.4765935 2.5125400
```

А еще данные удобно анализировать с помощью графиков

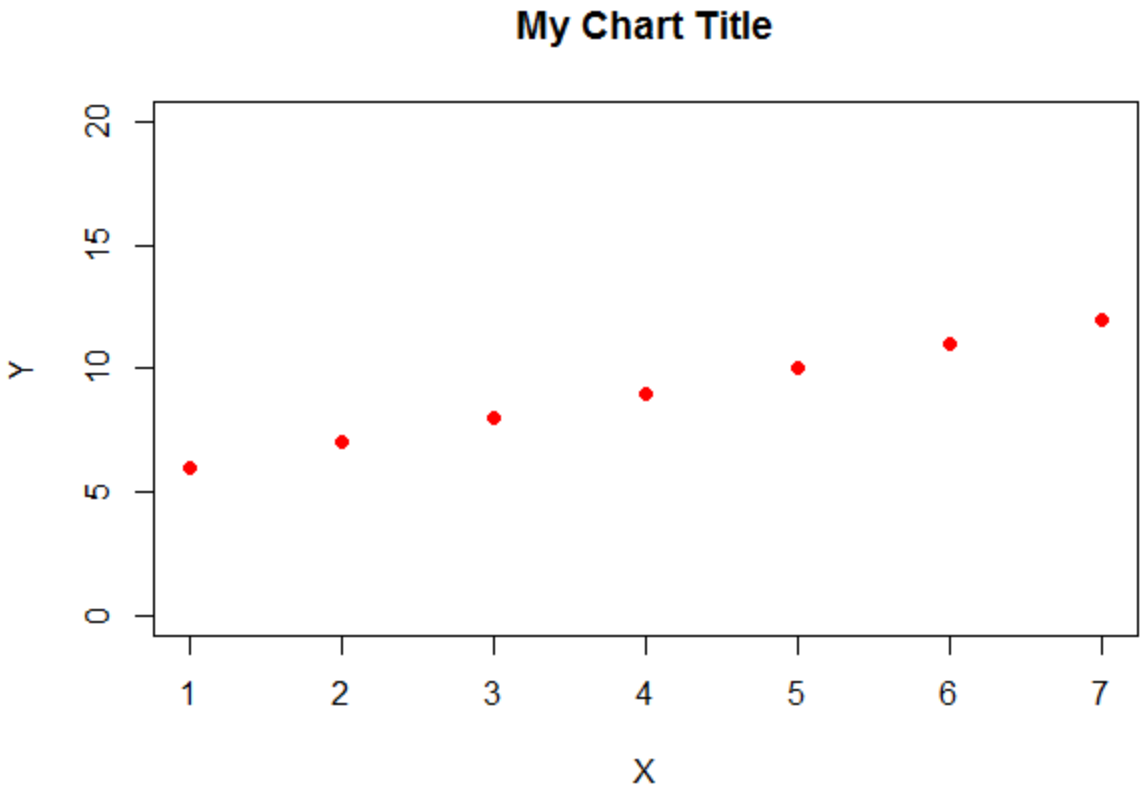
# Самый простой график

```
>x_data <- c(0:10)
>y_data <- x_data +5
>plot(x_data, y_data,  main = "My Chart Title", xlab ="X", ylab = "Y",
pch=16, col = "red")
```



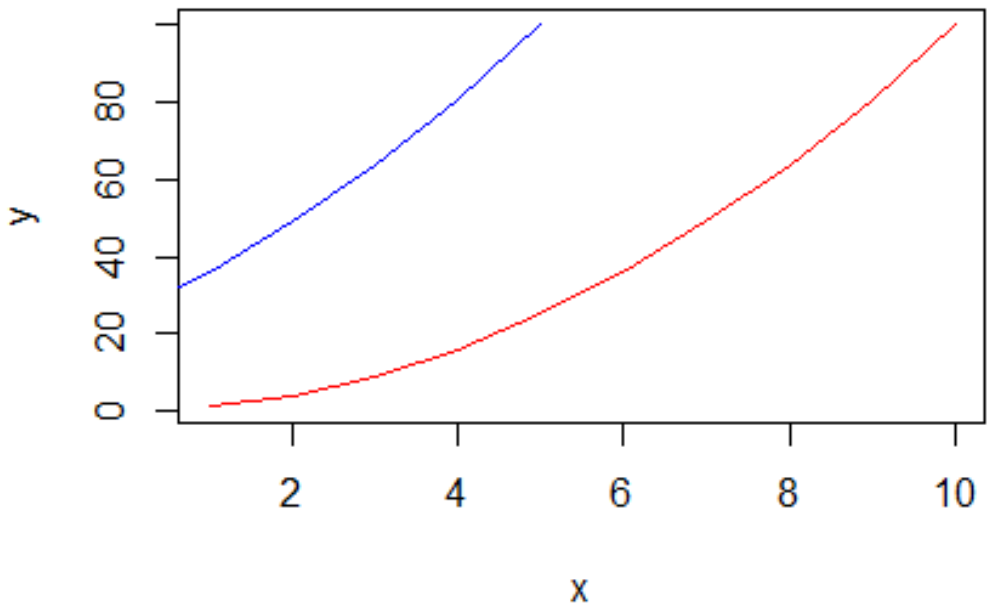
# Параметры xlim, ylim

>plot(x_data, y_data,  main = "My Chart Title", xlab ="X", ylab = "Y", pch=16, col = "red", xlim=c(1,7), ylim=c(0, 20))



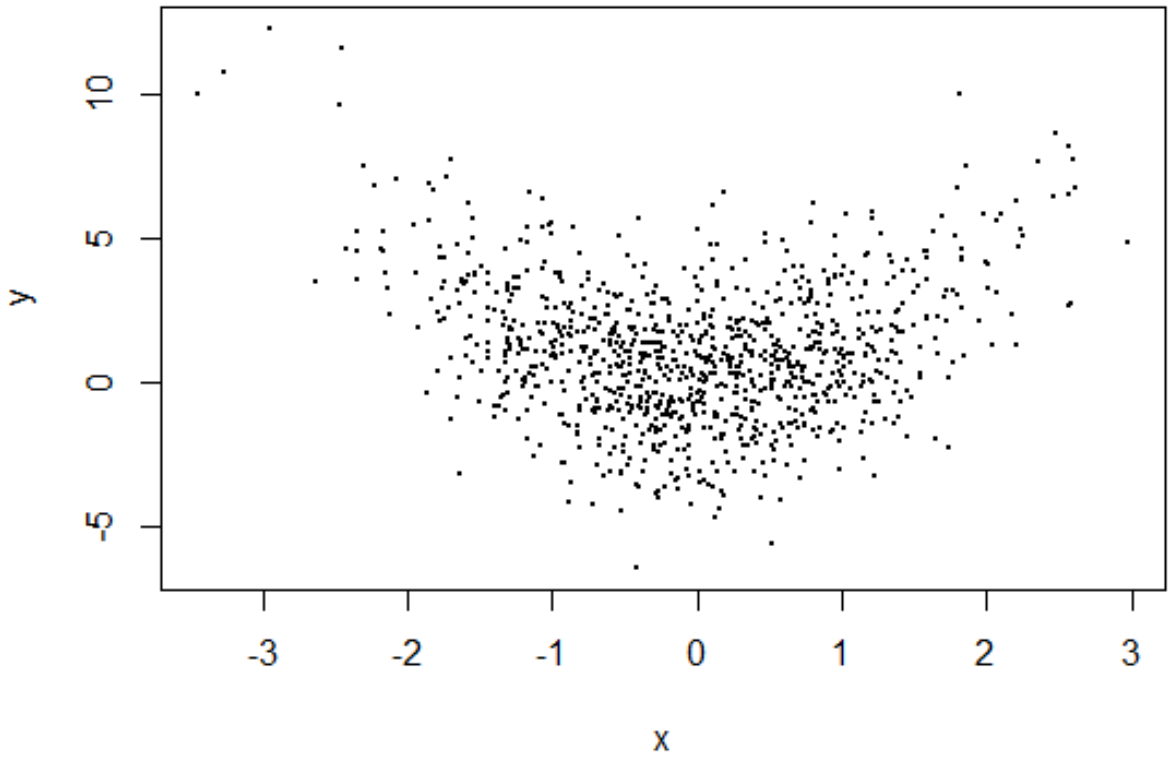
# Линии

```
> x <- 1:10
> y <- x*x
> z <- x-5
```



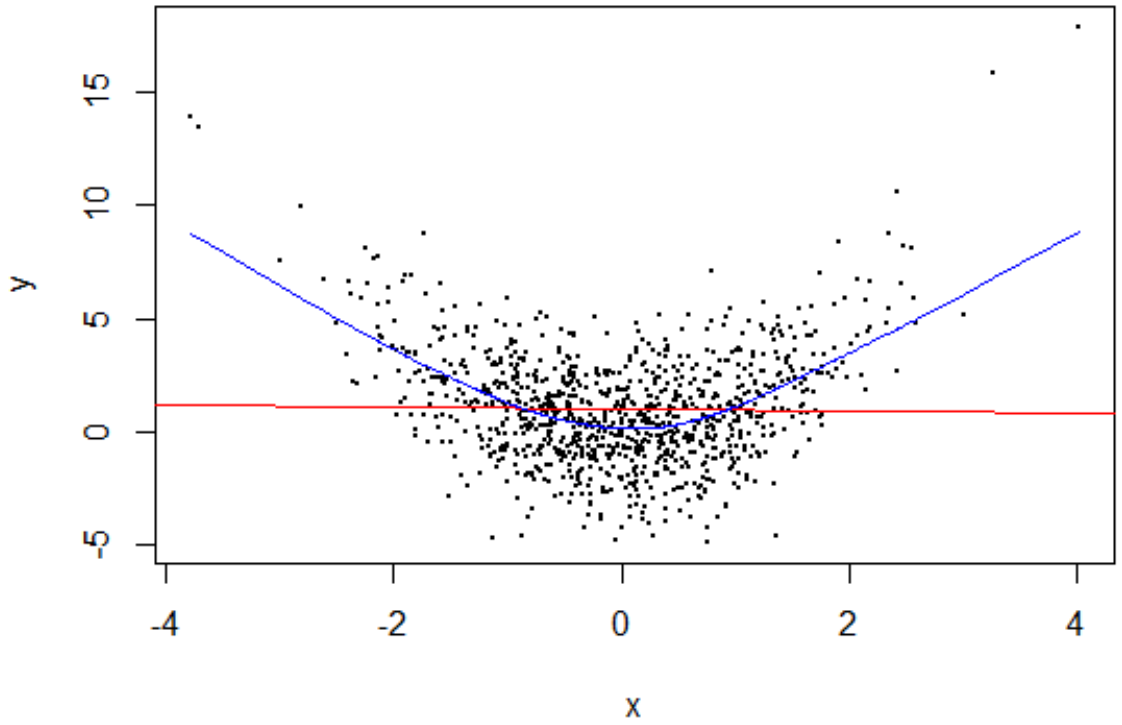

```
> plot(y ~ x, type= “l” , col =  ‘red')
> lines(y ~ z, col =  ‘blue')
```

# Scatterplots




```
> x<-rnorm(1000)
> y<-x*x + rnorm(1000, sd=2)
> plot(x, y, pch=19, cex=0.3)
```

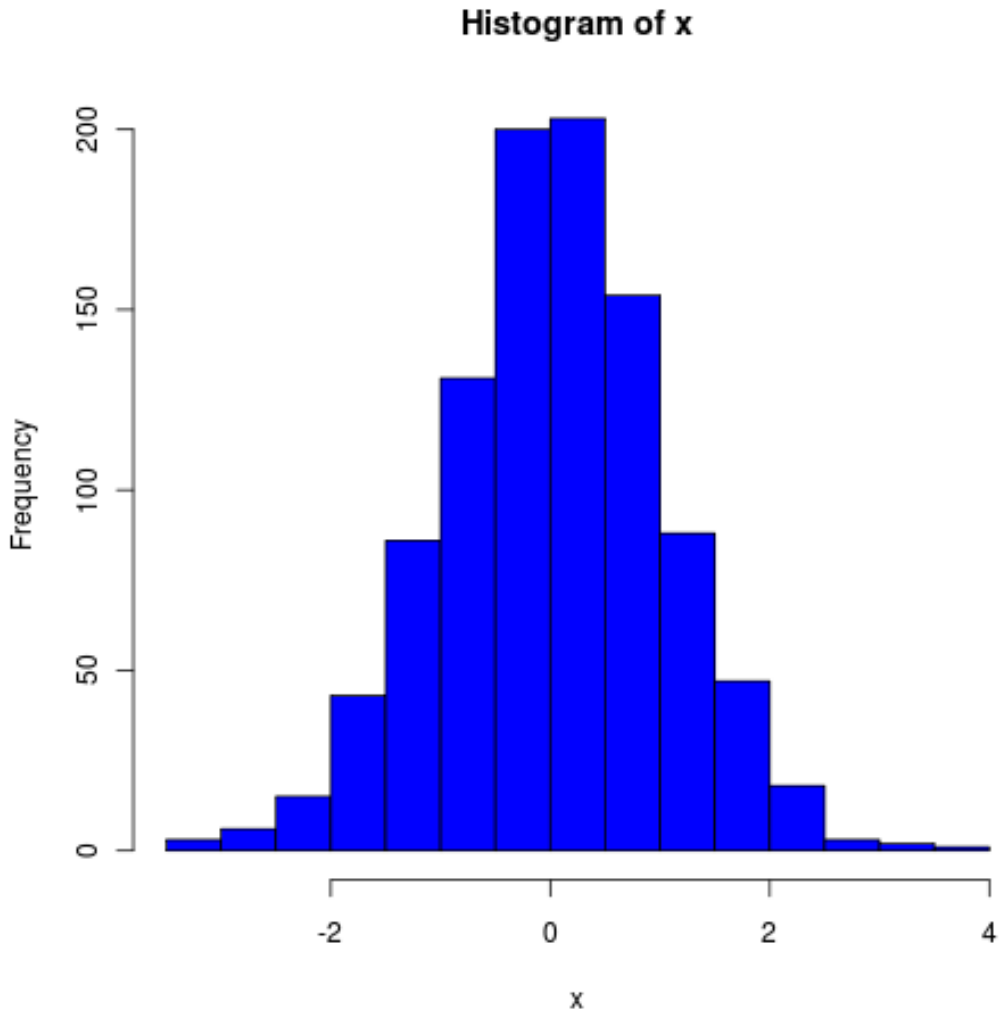
# Scatterplots: добавим линии



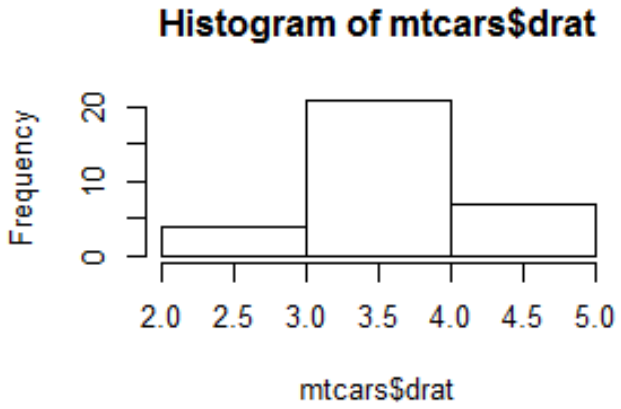

```
> abline(lm(y~x), col="red")
> lines(lowess(y~x), col="blue")
```

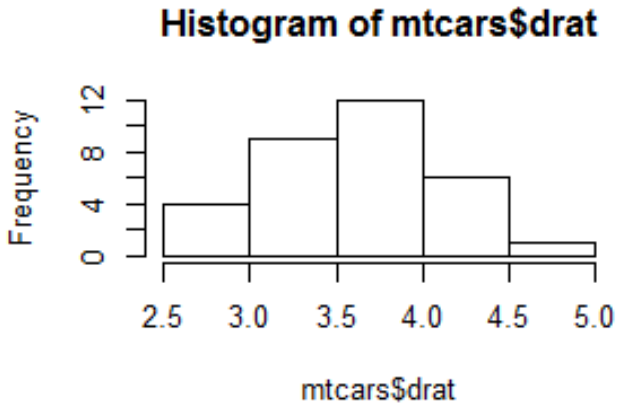
# Гистограммы

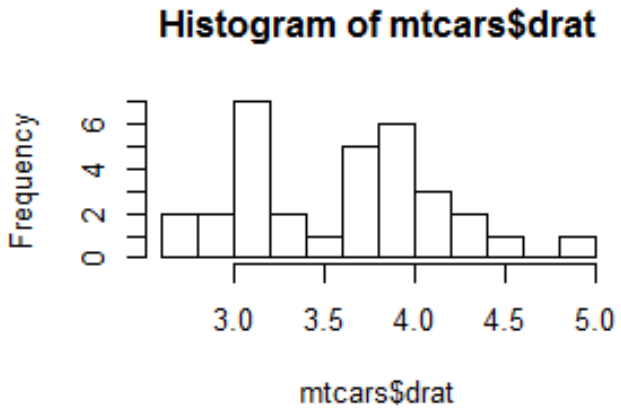
```
> x <- rnorm(1000)
> hist(x, col='blue')
```

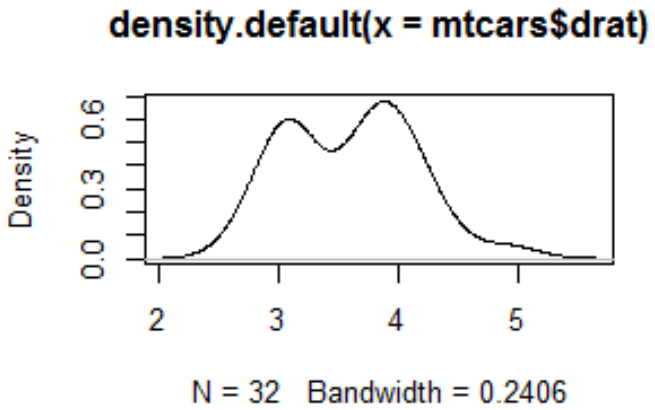


# Гистограммы

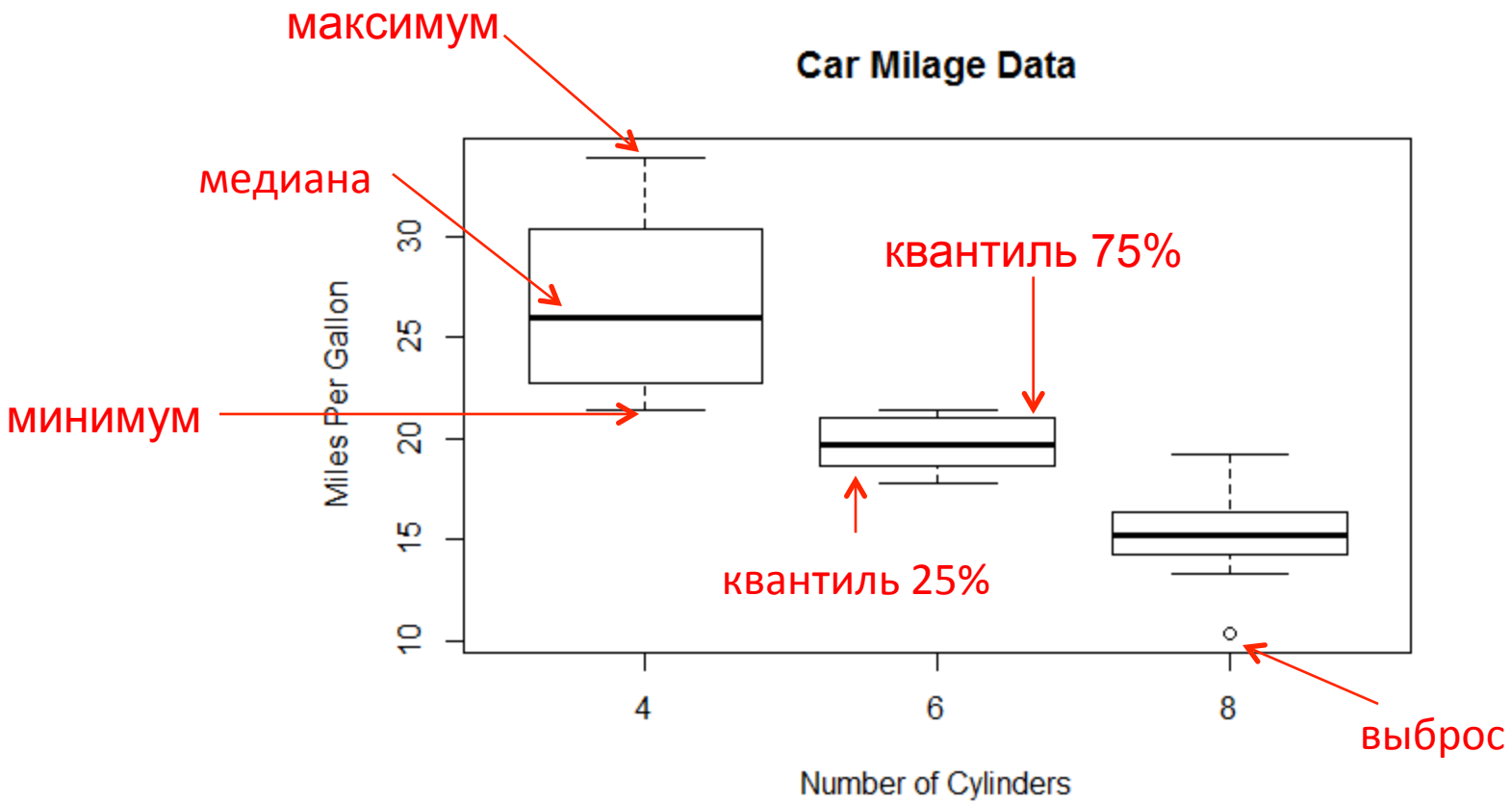


# Boxplots




> boxplot(mpg~cyl,data=mtcars, main="Car Milage Data", xlab="Number of Cylinders", ylab="Miles Per Gallon")

# Сохранение графика в файл

```
> png(file="Pictures/boxplot.png” , width=400, height=350, res=72)
> boxplot(x,y)
> dev.off()
```

Другие форматы:

| | |
|---|---|
| **pdf("mygraph.pdf")** | pdf file: для печати |
| **win.metafile("mygraph.wmf")** | windows metafile |
| **png("mygraph.png")** | png file: для веба |
| **jpeg("mygraph.jpg")** | jpeg file: не рекомендуем |
| **bmp("mygraph.bmp")** | bmp file |
| **postscript("mygraph.ps")** | postscript file |

# QUIZ TIME

?

Для воспроизводимости результата установите начальную точку для генератора псевдослучайных чисел с помощью команды set.seed(100)
С помощью случайной генерации из равномерного распределения создайте вектор x из 15 чисел.

Создайте вектор y, равный сумме вектора x и вектора, полученного генерацией из нормального распределения со средним 1 среднеквадратичным отклонением 0.2.

Нарисуйте точечную диаграмму зависимости y от x.

Сколько точек расположено выше регрессионной прямой?

Чему равно среднее вектора y, созданного в предыдущем задании?

Создайте вектор x, состоящий из первых 20 членов геометрической прогрессии со знаменателем √2 (делается командой sqrt(2)) и первым членам также равным √2. Для этого вам может понадобиться функция создания последовательности чисел seq и оператор возведения в степень ^: x^y означает x в степени y

Создайте еще один вектор y, в котором каждое значение y[i] равно максимуму из набора случайных (псевдослучайных) чисел, распределенных нормально с параметрами (0,1). При этом количество чисел в наборе должно быть равно значению x[i], округленному до целого числа.

Каков характер роста значений y в зависимости от x?

С помощью случайной генерации из равномерного распределения создайте вектор из 1000 значений. Нарисуйте гистограмму распределения чисел в векторе. Создайте еще один вектор из 1000 случайных равномерно распределенных значений. Сложите эти вектора и нарисуйте гистограмму распределения суммы двух независимых случайных величин, распределенных равномерно.

Повторите генерацию 1000 равномерно распределенных случайных значений и нарисуйте гистограмму распределения суммы уже трех независимых случайных величин.

По мере увеличения количества суммируемых случайных величин распределение должно все больше приближаться к нормальному.

# Data Frames

# Что такое data frame

- Структура данных: таблица из нескольких векторов (по столбцам), в разных столбцах могут быть данные разных типов

| | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mazda RX4 | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| Datsun 710 | 22.8 | 4 | 108.0 | 93 | 3.85 | 2.320 | 18.61 | 1 | 1 | 4 | 1 |
| Hornet 4 Drive | 21.4 | 6 | 258.0 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 | 1 |
| Hornet Sportabout | 18.7 | 8 | 360.0 | 175 | 3.15 | 3.440 | 17.02 | 0 | 0 | 3 | 2 |
| Valiant | 18.1 | 6 | 225.0 | 105 | 2.76 | 3.460 | 20.22 | 1 | 0 | 3 | 1 |
| Duster 360 | 14.3 | 8 | 360.0 | 245 | 3.21 | 3.570 | 15.84 | 0 | 0 | 3 | 4 |
| Merc 240D | 24.4 | 4 | 146.7 | 62 | 3.69 | 3.190 | 20.00 | 1 | 0 | 4 | 2 |
| Merc 230 | 22.8 | 4 | 140.8 | 95 | 3.92 | 3.150 | 22.90 | 1 | 0 | 4 | 2 |
| Merc 280 | 19.2 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.30 | 1 | 0 | 4 | 4 |
| Merc 280C | 17.8 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.90 | 1 | 0 | 4 | 4 |
| Merc 450SE | 16.4 | 8 | 275.8 | 180 | 3.07 | 4.070 | 17.40 | 0 | 0 | 3 | 3 |

# Как создать свой data frame?

```
> n <- c(2, 3, 5)
> s <- c("aa", "bb", "cc")
> b <- c(TRUE, FALSE, TRUE)
> df <- data.frame(n, s, b)
```

Или короче:

```
> df <- data.frame(n=c(2, 3, 5),
        s=c("aa", "bb", "cc"),
        b= c(TRUE, FALSE, TRUE))
```

# Основные команды

```
> df <- data.frame(n=c(2, 3, 5),  s=c("aa", "bb", "cc"), b=
c(TRUE, FALSE, TRUE))
> df
  n  s     b
1 2 aa  TRUE
2 3 bb FALSE
3 5 cc  TRUE
> df$n
[1] 2 3 5
> colnames(df)
[1] "n"  "s"  "b"
> rownames(df)
[1] "1"  "2"  "3"
> dim(df)
[1] 3 3
```

# Обращение к столбцу по имени, можно использовать tab!

# Важно, что это имена строк, а не числа!

# Использование data()

> mtcars

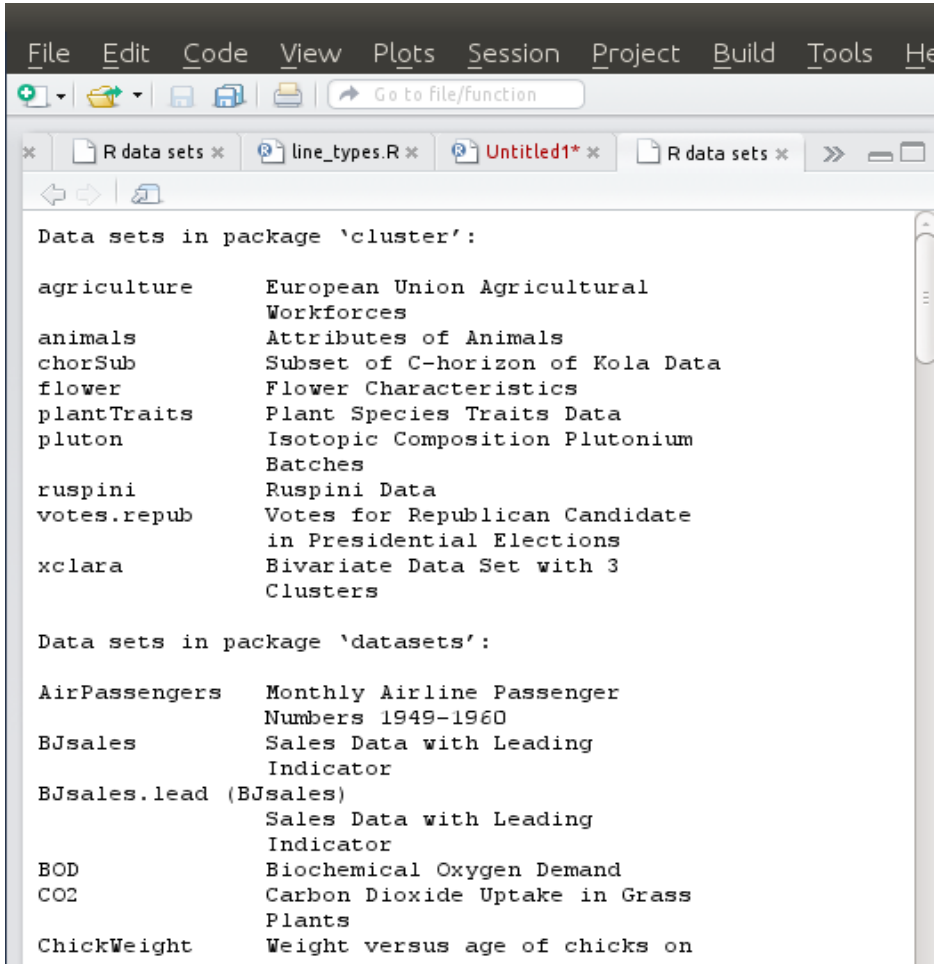
| | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mazda RX4 | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| Datsun 710 | 22.8 | 4 | 108.0 | 93 | 3.85 | 2.320 | 18.61 | 1 | 1 | 4 | 1 |
| Hornet 4 Drive | 21.4 | 6 | 258.0 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 | 1 |
| Hornet Sportabout | 18.7 | 8 | 360.0 | 175 | 3.15 | 3.440 | 17.02 | 0 | 0 | 3 | 2 |
| Valiant | 18.1 | 6 | 225.0 | 105 | 2.76 | 3.460 | 20.22 | 1 | 0 | 3 | 1 |
| Duster 360 | 14.3 | 8 | 360.0 | 245 | 3.21 | 3.570 | 15.84 | 0 | 0 | 3 | 4 |
| Merc 240D | 24.4 | 4 | 146.7 | 62 | 3.69 | 3.190 | 20.00 | 1 | 0 | 4 | 2 |
| Merc 230 | 22.8 | 4 | 140.8 | 95 | 3.92 | 3.150 | 22.90 | 1 | 0 | 4 | 2 |
| Merc 280 | 19.2 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.30 | 1 | 0 | 4 | 4 |
| Merc 280C | 17.8 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.90 | 1 | 0 | 4 | 4 |
| Merc 450SE | 16.4 | 8 | 275.8 | 180 | 3.07 | 4.070 | 17.40 | 0 | 0 | 3 | 3 |

*Командой data() можно посмотреть, какие выборки загружены для использования*

> data()

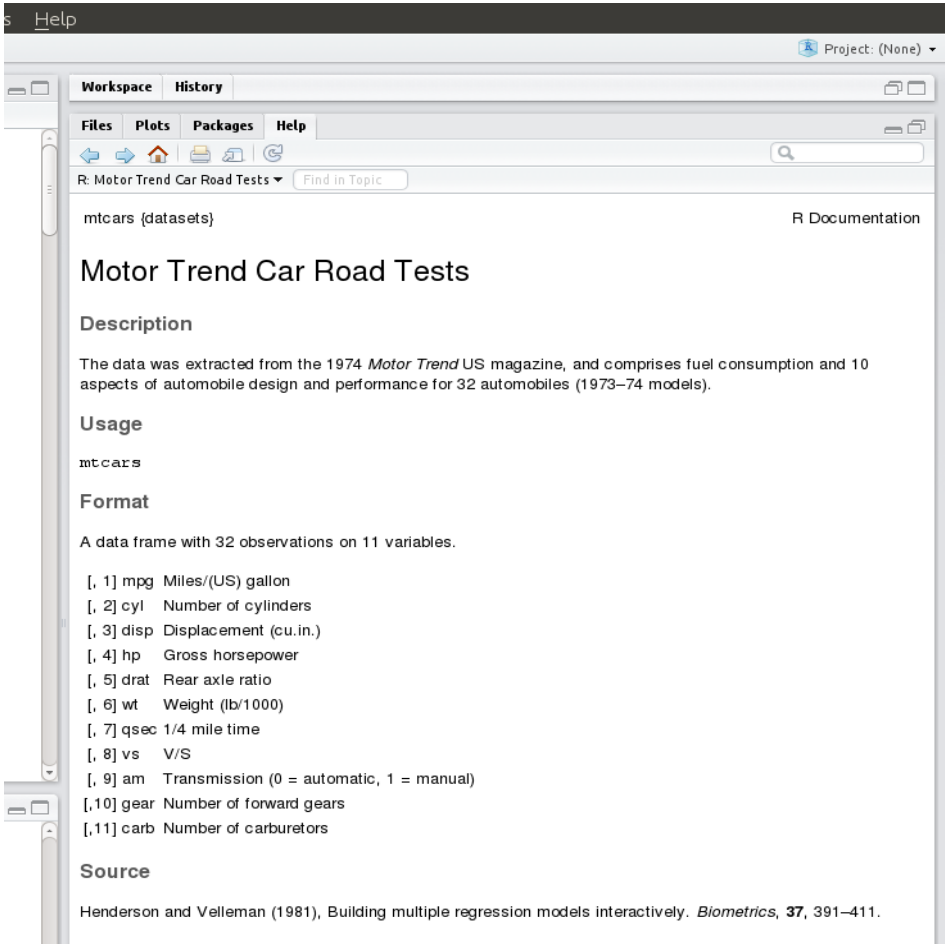
# Использование data()

> data()



# *Можно узнать о доступной выборке более подробно*

> ?mtcars



# Выбор строк, столбцов, ячеек

```
> mtcars[12,2]     # строка 12, столбец 2
[1] 8

> mtcars[8,]
           mpg cyl disp hp drat wt qsec vs am gear carb
Merc 240D 24.4 4 146.7 62 3.69 3.19 20 1 0 4 2

> mtcars[1:3,]     # строки 1 - 3, все столбцы
           mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4     21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710    22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
```

# Выбор строк, столбцов, ячеек

```
> mtcars[,2]          # все строки, столбец 2
 [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 4 8 6 8 4
```

```
> mtcars[c(1,13),]  # строки 1 и 13, все столбцы
           mpg cyl  disp  hp drat   wt  qsec vs am gear carb
Mazda RX4   21.0   6 160.0 110 3.90 2.62 16.46  0  1    4    4
Merc 450SL 17.3   8 275.8 180 3.07 3.73 17.60  0  0    3    3
```

```
> mtcars[c(1,3,7,13),1]
# строки 1, 3, 7 и 13, столбец 1
[1] 21.0 22.8 14.3 17.3
```

# Добавить столбец

```
> dim(mtnew)
[1] 33 11
> num<-1:33
> mtnew<-cbind(mtnew, num)          #добавляем столбец
> mtnew[30:33,]
                mpg cyl disp  hp drat   wt  qsec vs am gear carb num
Ferrari Dino   19.7   6  145 175 3.62 2.77 15.50  0  1    5    6  30
Maserati Bora  15.0   8  301 335 3.54 3.57 14.60  0  1    5    8  31
Volvo 142E     21.4   4  121 109 4.11 2.78 18.60  1  1    4    2  32
Lada           21.0   6  150 120 4.00 2.50 16.46  1  1    4    4  33
```

# Добавить строку

```
> mtnew<-mtcars
> dim(mtnew)
[1] 32 11
> mtnew[1,]
          mpg cyl disp  hp drat   wt  qsec vs am gear carb
Mazda RX4  21   6  160 110  3.9 2.62 16.46  0  1    4    4
> newcar<-c(21, 6, 150, 120, 4.0, 2.5, 16.46, 1, 1, 4, 4)#работает только если все данные одного типа!!!!
> newcar<-data.frame(mpg=21, cyl=4, disp=100, hp=80, drat=1, wt=2, qsec=16, vs=1,am=0, gear=4, carb=1) # data.frame из 1 строки
> mtnew<-rbind(mtnew, newcar)      #добавляем строку
> rownames(mtnew)[33]<-"Lada"      #присваиваем ей имя
> mtnew[30:33,]
               mpg cyl disp  hp drat   wt  qsec vs am gear carb
Ferrari Dino  19.7   6  145 175 3.62 2.77 15.50  0  1    5    6
Maserati Bora 15.0   8  301 335 3.54 3.57 14.60  0  1    5    8
Volvo 142E    21.4   4  121 109 4.11 2.78 18.60  1  1    4    2
Lada          21.0   6  150 120 4.00 2.50 16.46  1  1    4    4
```

# Логические условия и order

```
> mtcars1 <- mtcars[mtcars$cyl>4 & mtcars$cyl<8,]
> mtcars1
```

| | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mazda RX4 | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| Hornet 4 Drive | 21.4 | 6 | 258.0 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 | 1 |
| Valiant | 18.1 | 6 | 225.0 | 105 | 2.76 | 3.460 | 20.22 | 1 | 0 | 3 | 1 |
| Merc 280 | 19.2 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.30 | 1 | 0 | 4 | 4 |
| Merc 280C | 17.8 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.90 | 1 | 0 | 4 | 4 |
| Ferrari Dino | 19.7 | 6 | 145.0 | 175 | 3.62 | 2.770 | 15.50 | 0 | 1 | 5 | 6 |

```
> mtcars1[order(mtcars1$drat),]
```

| | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Valiant | 18.1 | 6 | 225.0 | 105 | 2.76 | 3.460 | 20.22 | 1 | 0 | 3 | 1 |
| Hornet 4 Drive | 21.4 | 6 | 258.0 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 | 1 |
| Ferrari Dino | 19.7 | 6 | 145.0 | 175 | 3.62 | 2.770 | 15.50 | 0 | 1 | 5 | 6 |
| Mazda RX4 | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| Merc 280 | 19.2 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.30 | 1 | 0 | 4 | 4 |
| Merc 280C | 17.8 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.90 | 1 | 0 | 4 | 4 |

# Факторы

Используются для представления категориальных данных (да/нет, низкий/средний/высокий, мужчина/женщина…)

```
> f <- factor(c("yes", "yes", "no", "yes", "no"))
> f
[1] yes yes no yes no
Levels: no yes

> levels(f)       # возможные значения в факторе
[1] "no" "yes”
> levels(f) <- c(levels(f), "maybe")

> table(f)
f
 no  yes  maybe
  2    3      0
```

# Факторы

Уровни можно упорядочивать при создании фактора
(может быть важно в линейной регрессии):

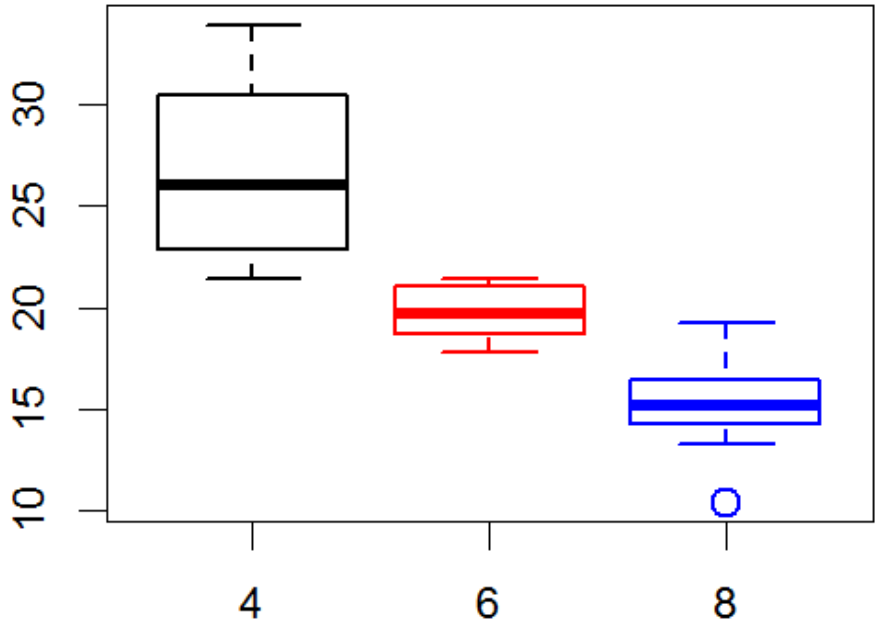
```
> f <- factor(c("yes", "yes", "no", "yes",
"no"), levels = c("yes", "no"))
```

```
> f
 [1] yes yes no yes no
Levels: yes no
```

# Факторы

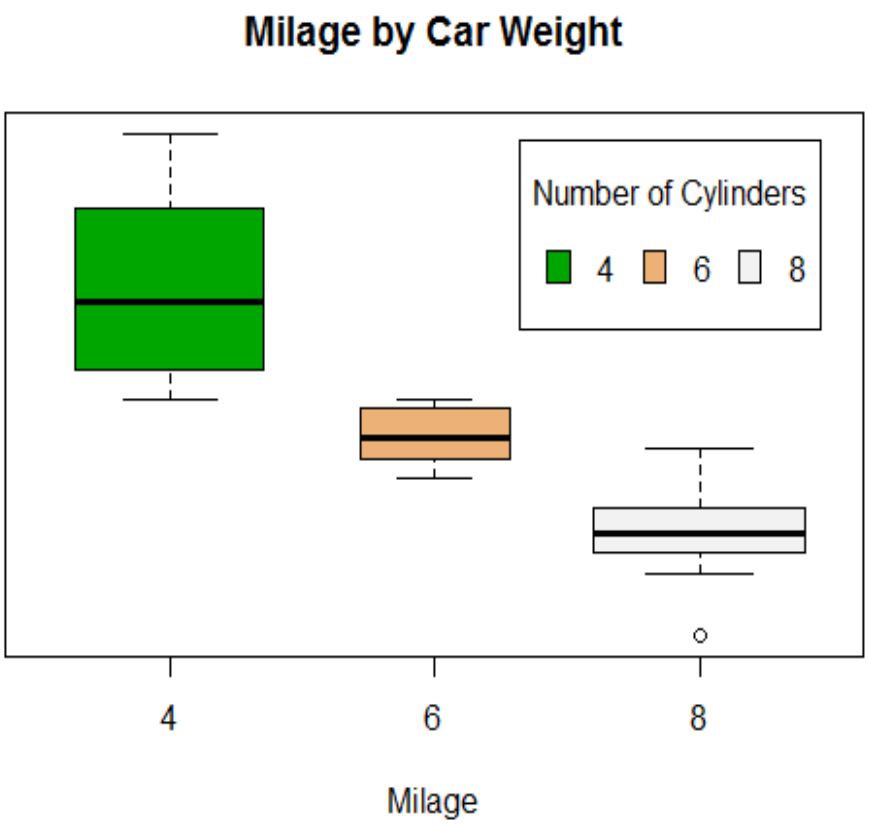
Разбиение вектора по фактору:

- `boxplot(mtcars$mpg ~ mtcars$cyl)`




```
mpg: 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 …
cyl:  6    6    4    6    8    6    8    4   …
```

# Параметр legend

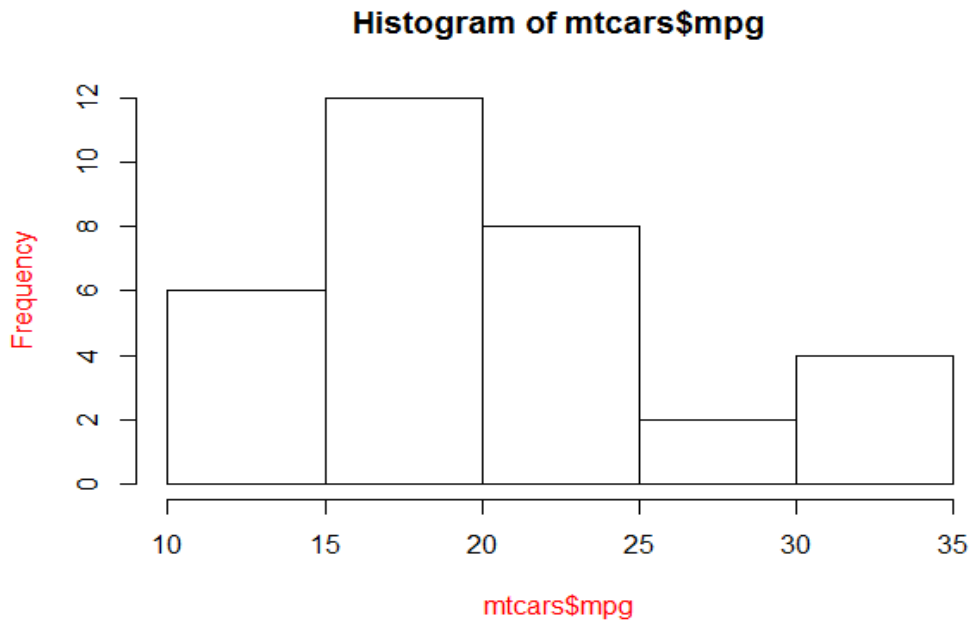



```
> boxplot(
mtcars$mpg~mtcars$cyl,
main="Milage by Car Weight",
yaxt="n", xlab="Milage",
col=terrain.colors(3), varwidth=T)
> legend("topright", inset=.05,
title="Number of Cylinders",
c("4","6","8"), fill=terrain.colors(3),
horiz=TRUE)
```

# Графический параметр par()

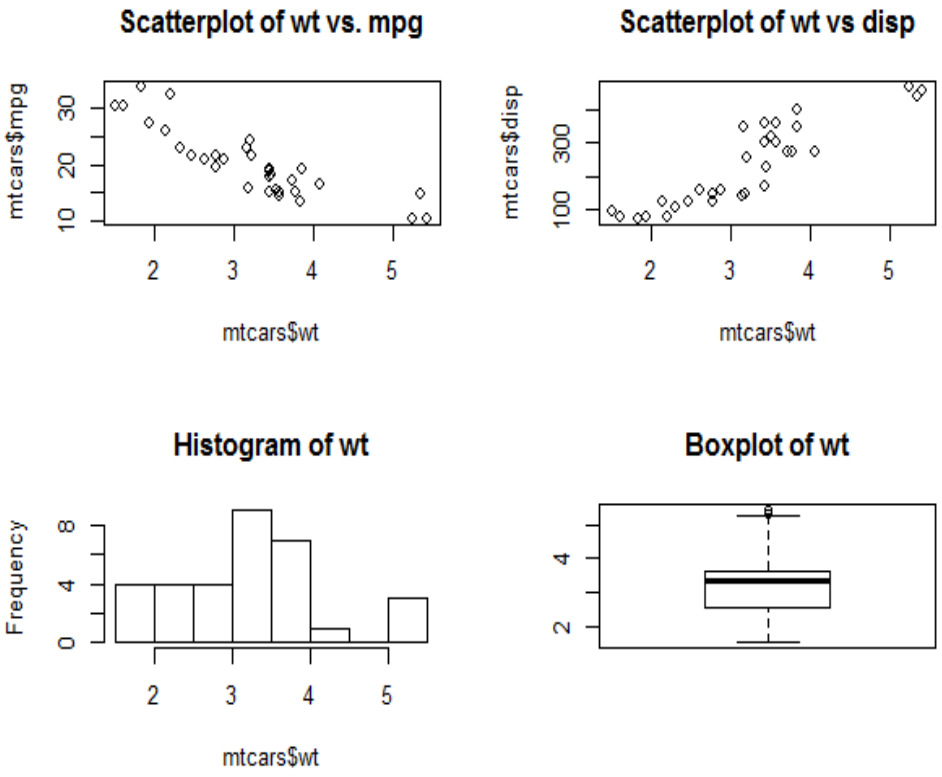
```
> par()     # просмотреть текущие значения графических параметров
>old_par<-par( no.readonly = TRUE ) #прежде чем менять настройки, рекомендуем сохранить старые
> par(col.lab="red") # сделать красными подписи к осям
> hist(mtcars$mpg)  # график рисуется с новыми настройками
```

!




```
>par(old_par) #восстанавливаем старые настройки
```

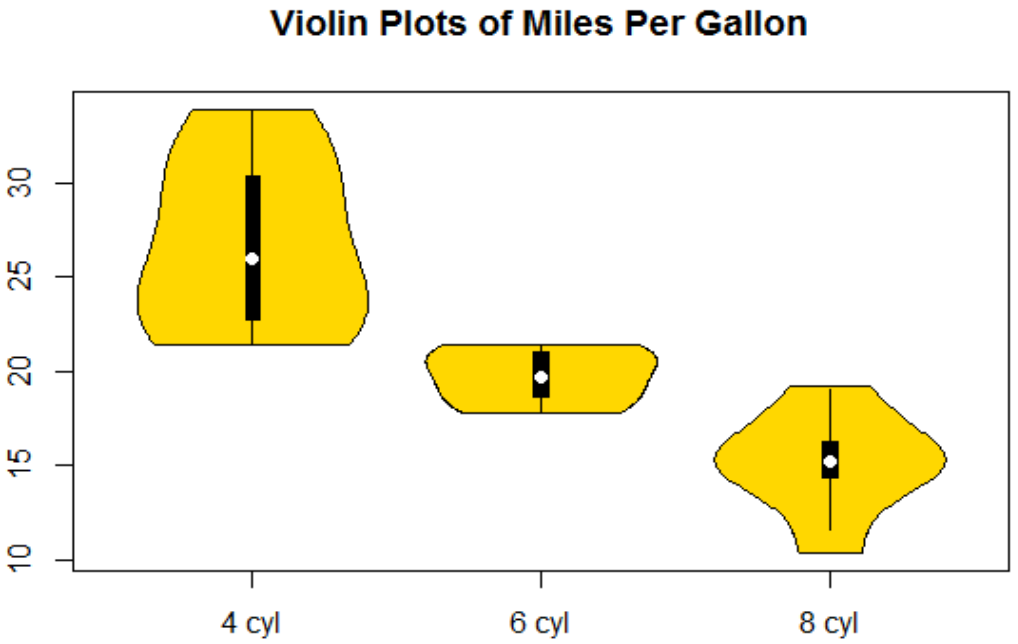
# Комбинация графиков




```
> par(mfrow=c(2,2))
> plot(mtcars$wt,mtcars$mpg,
main="Scatterplot of wt vs. mpg")
> plot(mtcars$wt,mtcars$disp,
main="Scatterplot of wt vs disp")
> hist(mtcars$wt, main="Histogram
of wt")
> boxplot(mtcars$wt,
main="Boxplot of wt")
```

# Violin Plot: комбинация boxplot и графика плотности распределения

*«The violin plot is like the lovechild between a density plot and a box-and-whisker plot.»*




```
> library(vioplot)
> x1 <- mtcars[mtcars$cyl==4,]$mpg
> x2 <- mtcars[mtcars$cyl==6,]$mpg
> x3 <- mtcars[mtcars$cyl==8,]$mpg
> vioplot(x1, x2, x3, names=c("4 cyl", "6 cyl",
"8 cyl"), col="gold")
title("Violin Plots of Miles Per Gallon")
```

# QUIZ TIME

?

Постройте график плотности распределения веса для данных mtcars (mtcars$wt).

Какое из утверждений является верным?
a. распределение является унимодальным
b. распределение является бимодальным
c. распределение является равномерным

Постройте boxplot для времени разгона машин (1/4 mile time) в зависимости от типа коробки передач.

Какие утверждения являются верными?
a. Распределение для машин с автоматической коробкой передач имеет выброс
b. Медиана распределения для машин с автоматической коробкой передач равна 75% квантили распределения для машин с механической коробкой передач
c. Машин с автоматической коробкой передач представлено больше, чем машин с механической коробкой передач

Отсортируйте данные о состоянии воздуха в Нью-Йорке (airquality) по месяцам и температуре воздуха  (в порядке возрастания).

Какая скорость ветра указана в третьей по порядку строке?

# Работа с файлами

# Работа с файлами: основные функции

| Чтение | Запись | Применение |
|---|---|---|
| ***read.table*** | ***write.table*** | Чтение/запись табулированных текстовых файлов |
| ***read.csv*** | ***write.csv*** | Чтение/запись файлов в формате CSV |
| ***readLines*** | ***writeLines*** | Чтение/запись текстовых файлов по строкам |
| ***load*** | ***save*** | Загрузка/сохранение объектов R из/в бинарные файлы (.RData) |

# Работа с файлами: рабочая директория

Узнать рабочую директорию:

```
> getwd()
```

Поменять рабочую директорию:

```
> setwd("Week3")  # путь указан относительно рабочей директории!
```

Узнать список файлов в рабочей директории

```
> dir()
```

Узнать список файлов в указанной директории

```
> dir(”/Users/anna/FBB/R/")
```

**В RStudio:**

закладка Files (справа внизу) -> выбрать нужную директорию -> More -> Set As Working Directory

# Работа с файлами: *read.table*

- Читает файл с разделителями
- Возвращает ***data.frame***

```
> students <- read.table("FBBRStudents.tab",sep="\t", header=T)

> students[101:102,]
          Name       Faculty        Level Year
101 Широкий В. Р.    химический  специалитет    4
102 Базылев С. С. биологический  бакалавриат    1
```

# Работа с файлами: *read.table*

Основные аргументы:

- file – имя файла или соединение (connection)
- header – есть ли в файле заголовок? (по умолчанию, FALSE)
- sep – разделитель полей (колонок) (по умолчанию, пробел)
- colClasses – вектор с названиями классов колонок
- nrows – количество строчек, которые нужно прочитать
- skip – количество строчек, которые нужно пропустить
- comment.char – знак комментариев
- stringsAsFactors – преобразовывать строковые поля в фактор? (по умолчанию, TRUE)

# Работа с файлами: *read.table*

```
> students<-read.table("FBBRStudents.tab",sep="\t",header=T,
+ colClasses = c("character","factor","factor","integer"))

> str(students)
'data.frame': 141 obs. of 4 variables:
 $ Name : chr "Антонов С. В." "Дмитриев Д. И." "Золотов И.
А." "Иванова Т. В." ...
 $ Faculty: Factor w/ 10 levels "биологический",..: 3 3 3 3
3 3 3 3 3 3 ...
 $ Level : Factor w/ 3 levels "бакалавриат",..: 3 3 3 3 3 3
3 3 3 3 ...
 $ Year : int 3 3 3 3 4 4 4 4 4 4 ...
```

# Работа с файлами: *read.csv, write.csv, readLines*

- read.csv – то же, что read.table, но с другими дефолтными значениями параметров (header=TRUE, sep=“,”)
- write.csv:

```
> write.csv(students, "FBBRStudents.csv")
```

- readLines:

```
> lines <- readLines("FBBRStudents.txt“,3)

> lines
[1] "Name\tFaculty\tLevel\tYear"
[2] "Антонов С. В.\tмеханико-математический\tспециалитет\t3"
[3] "Дмитриев Д. И.\tмеханико-математический\tспециалитет\t3"
```

# Работа с файлами: *save, load*

Сохраняем объекты students и lines в файл:

```
> save(students, lines, file="Students.RData”)
```

Удаляем все объекты из рабочего пространства:

```
> rm(list=ls())
```

```
> ls()
character(0)
```

Загружаем объекты из файла:

```
> load("Students.RData")
```

```
> ls()
[1] "lines" "students“ # объекты появляются в
                        # рабочем пространстве
```

# Соединения

- file – открывает соединение с файлом
- gzfile, bzfile – открывает соединение с архивированным файлом
- url – открывает соединение с веб-страницей

```
> con <- file("FBBRStudents.txt", "r")

> readLines(con, 1)

[1] "Name\tFaculty\tLevel\tYear"

> readLines(con, 1)

[1] "Антонов С. В.\tмеханико-математический\tспециалитет\t3"

> close(con)

> con <- gzfile("FBBRStudents.gz")

> read.csv(con, nrow=2)

X Name Faculty Level Year
1 1 Антонов С. В. механико-математический специалитет 3
2 2 Дмитриев Д. И. механико-математический специалитет 3
> close(con)
```

# Пакеты

(устанавливаем в R Studio)

# Красивые графики

# Как делать красивые и эффективные графики

Согласно Naomi Robbins, эффективные графики «улучшают понимание данных». Они не должны сбивать с толку!

Что мы хотим:

- сравнивать данные между собой
- выявлять тренды

# Почему не нужно использовать pie charts

Самый ужасный тип графика, но при этом потрясающе популярный!

http://www.richardhollins.com/blog/why-pie-charts-suck/

“The only worse design than a pie chart is several of them.”

This animation created by Darkhorse Analytics illustrates how communication can be greatly enhanced by eliminating clutter and de-emphasizing supporting elements. Every aspect of a figure should be there on a “need to have it” basis.


http://stat545-ubc.github.io/img/less-is-more-darkhorse-analytics.gif

from Wickham's *Tidy Data*

In **tidy data**:

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

## messy

| | treatmenta | treatmentb |
|---|---|---|
| John Smith | — | 2 |
| Jane Doe | 16 | 11 |
| Mary Johnson | 3 | 1 |

| | John Smith | Jane Doe | Mary Johnson |
|---|---|---|---|
| treatmenta | — | 16 | 3 |
| treatmentb | 2 | 11 | 1 |

## tidy

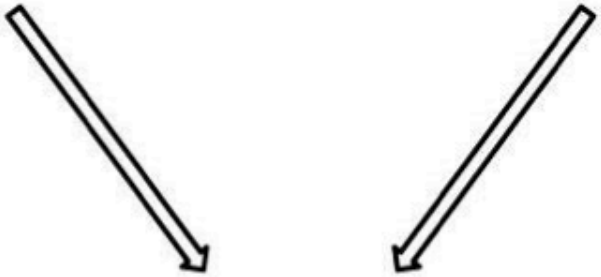
| name | trt | result |
|---|---|---|
| John Smith | a | — |
| Jane Doe | a | 16 |
| Mary Johnson | a | 3 |
| John Smith | b | 2 |
| Jane Doe | b | 11 |
| Mary Johnson | b | 1 |

https://github.com/jennybc/ggplot2-tutorial/tree/master/ggplot2-tutorial-slides

from White et al's *Nine simple ways* ...

| | Habitat | | |
|---|---|---|---|
| Species | X | Y | Z |
| A | 0 | 3 | 0 |
| B | 1 | 0 | 2 |

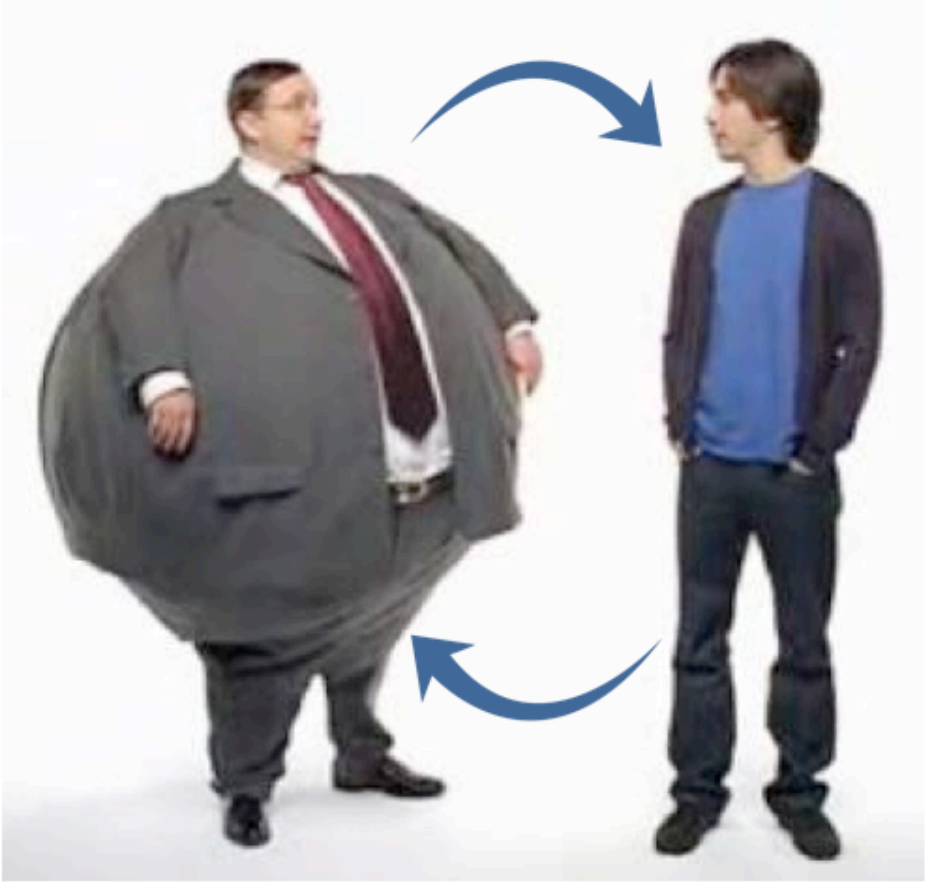
| Species | HabitatX | HabitatY | HabitatZ |
|---|---|---|---|
| A | 0 | 3 | 0 |
| B | 1 | 0 | 2 |



| Species | Habitat | Abundance |
|---|---|---|
| A | Y | 3 |
| B | X | 1 |
| B | Z | 2 |

https://github.com/jennybc/ggplot2-tutorial/tree/master/ggplot2-tutorial-slides

# reshape your data

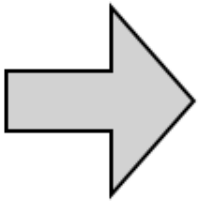


data has a tendency to get shorter and wider, but tall and thin often better for analysis + visualization

from Wickham's *Tidy Data*
see also reshape2

# reshape2::melt
# tidyr::gather

| row | a | b | c |
|---|---|---|---|
| a | 1 | 4 | 7 |
| b | 2 | 5 | 8 |
| c | 3 | 6 | 9 |



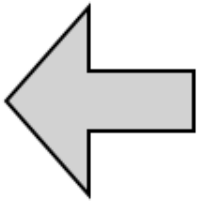
| row | column | value |
|---|---|---|
| a | a | 1 |
| b | a | 2 |
| c | a | 3 |
| a | b | 4 |
| b | b | 5 |
| c | b | 6 |
| a | c | 7 |
| b | c | 8 |
| c | c | 9 |

https://github.com/jennybc/ggplot2-tutorial/tree/master/ggplot2-tutorial-slides

from Wickham's *Tidy Data*
see also reshape2

# reshape2::cast
# tidyr::spread

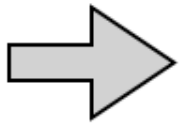
| row | a | b | c |
|---|---|---|---|
| a | 1 | 4 | 7 |
| b | 2 | 5 | 8 |
| c | 3 | 6 | 9 |



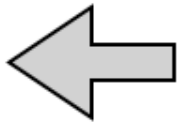
| row | column | value |
|---|---|---|
| a | a | 1 |
| b | a | 2 |
| c | a | 3 |
| a | b | 4 |
| b | b | 5 |
| c | b | 6 |
| a | c | 7 |
| b | c | 8 |
| c | c | 9 |

gather

| row | a | b | c |
|---|---|---|---|
| a | 1 | 4 | 7 |
| b | 2 | 5 | 8 |
| c | 3 | 6 | 9 |



| row | column | value |
|---|---|---|
| a | a | 1 |
| b | a | 2 |
| c | a | 3 |
| a | b | 4 |
| b | b | 5 |
| c | b | 6 |
| a | c | 7 |
| b | c | 8 |
| c | c | 9 |



spread

typical usage pattern:

gather to facilitate analysis and visualization

spread to make compact tables that are nicer for eyeballs

https://github.com/jennybc/ggplot2-tutorial/tree/master/ggplot2-tutorial-slides

# Пакет reshape2

```
install.packages("reshape2")
library(“reshape2”)

names(airquality) <- tolower(names(airquality))
head(airquality)

aql <- melt(airquality)
head(aql)
```

# Контролируем, какие переменные являются id

```
aql <- melt(airquality, id.vars = c("month", "day"),
  variable.name = "climate_variable",
  value.name = "climate_value")
head(aql)
```

# И обратно…

```
aql <- melt(airquality, id.vars = c("month", "day"))
aqw <- dcast(aql, month + day ~ variable)
head(aqw)
```

# ggplot2

## Author

ggplot2 was developed by Hadley Wickham, assistant professor of statistics at Rice University, Houston. In July 2010 the latest stable release (Version 0.8.8) was published.


Hadley Wickham
Dobelman Family Junior Chair
Statistics, Rice University
6100 Main St MS#138
Houston TX 77005-1827

February 3, 2010

515 450 8171
hadley@rice.edu
http://had.co.nz


2008 Ph.D. (Statistics), Iowa State University, Ames, IA. “Practical tools for exploring data and models.”

2004 M.Sc. (Statistics), First Class Honours, The University of Auckland, Auckland, New Zealand.

2002 B.Sc. (Statistics, Computer Science), First Class Honours, The University of Auckland, Auckland, New Zealand.

1999 Bachelor of Human Biology, First Class Honours, The University of Auckland, Auckland, New Zealand.

http://www.ceb-institute.org/bbs/wp-content/uploads/2011/09/handout_ggplot2.pdf

```
### устанавливаем и загружаем пакет
install.packages("ggplot2")
library("ggplot2")
```

**data,** in data.frame form

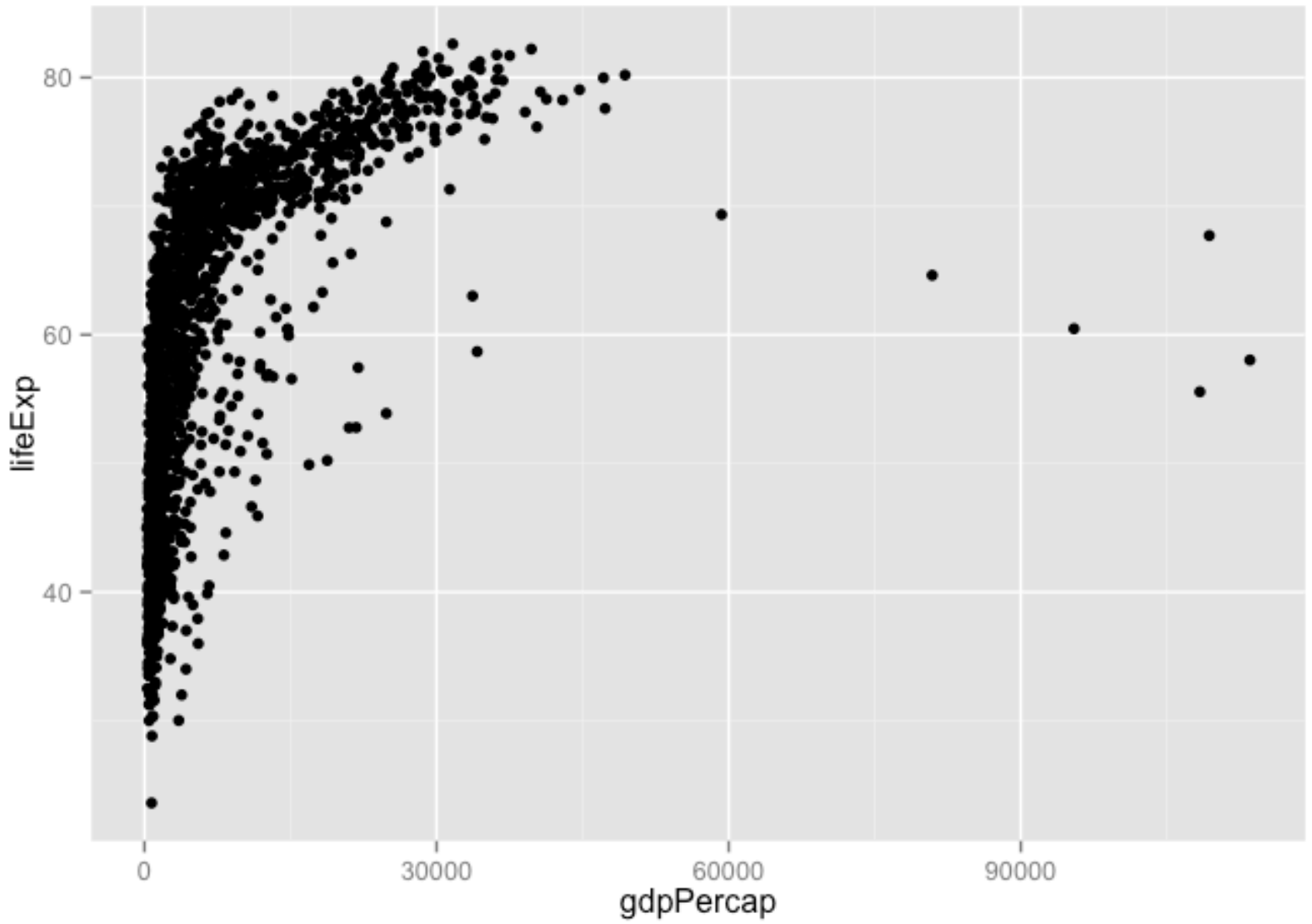
**aesthetic**: map variables into properties people can perceive visually ... position, color, line type?

**geom**: specifics of what people see ... points? lines?

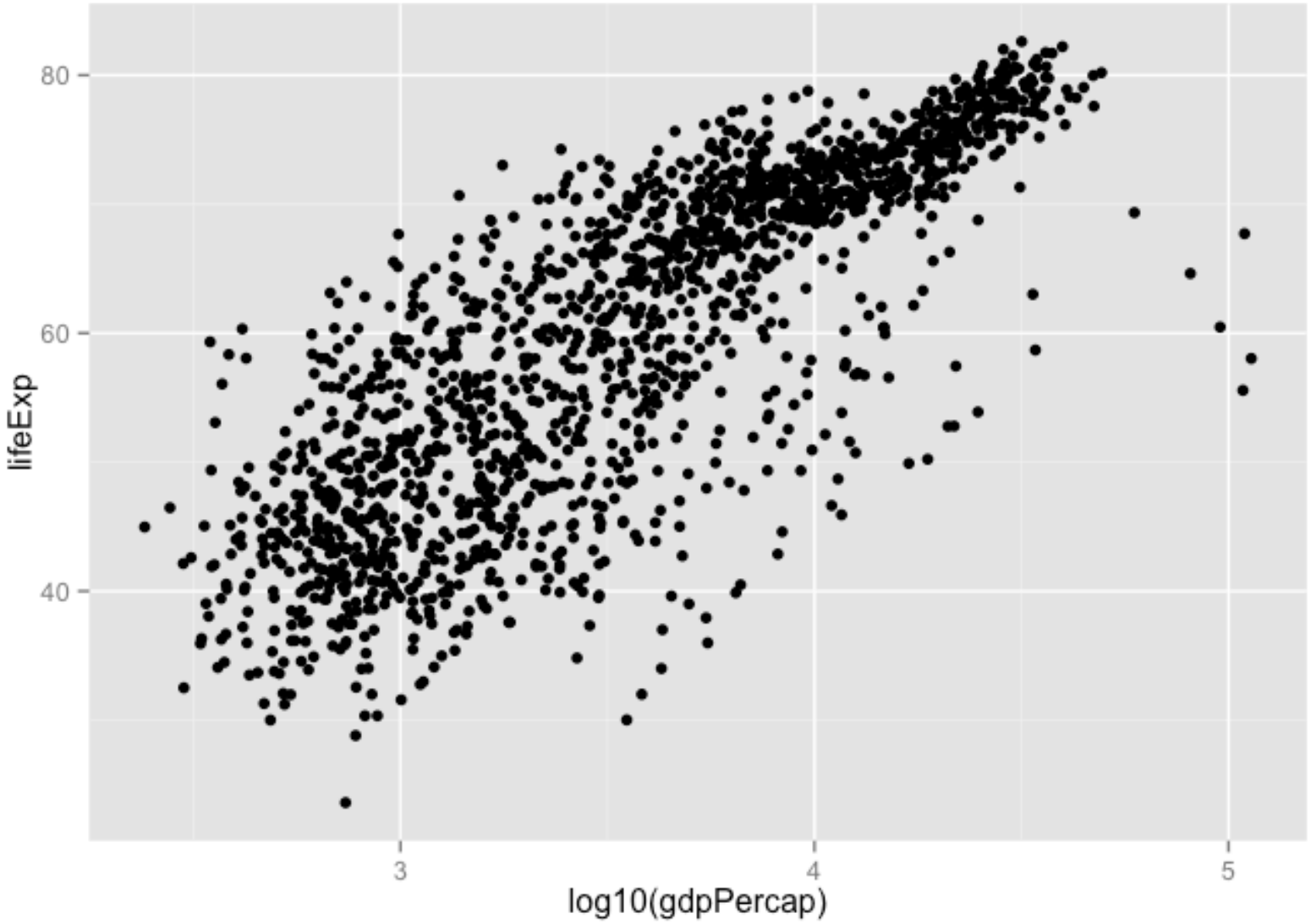
**scale**: map data values into "computer" values

**stat**: summarization/transformation of data

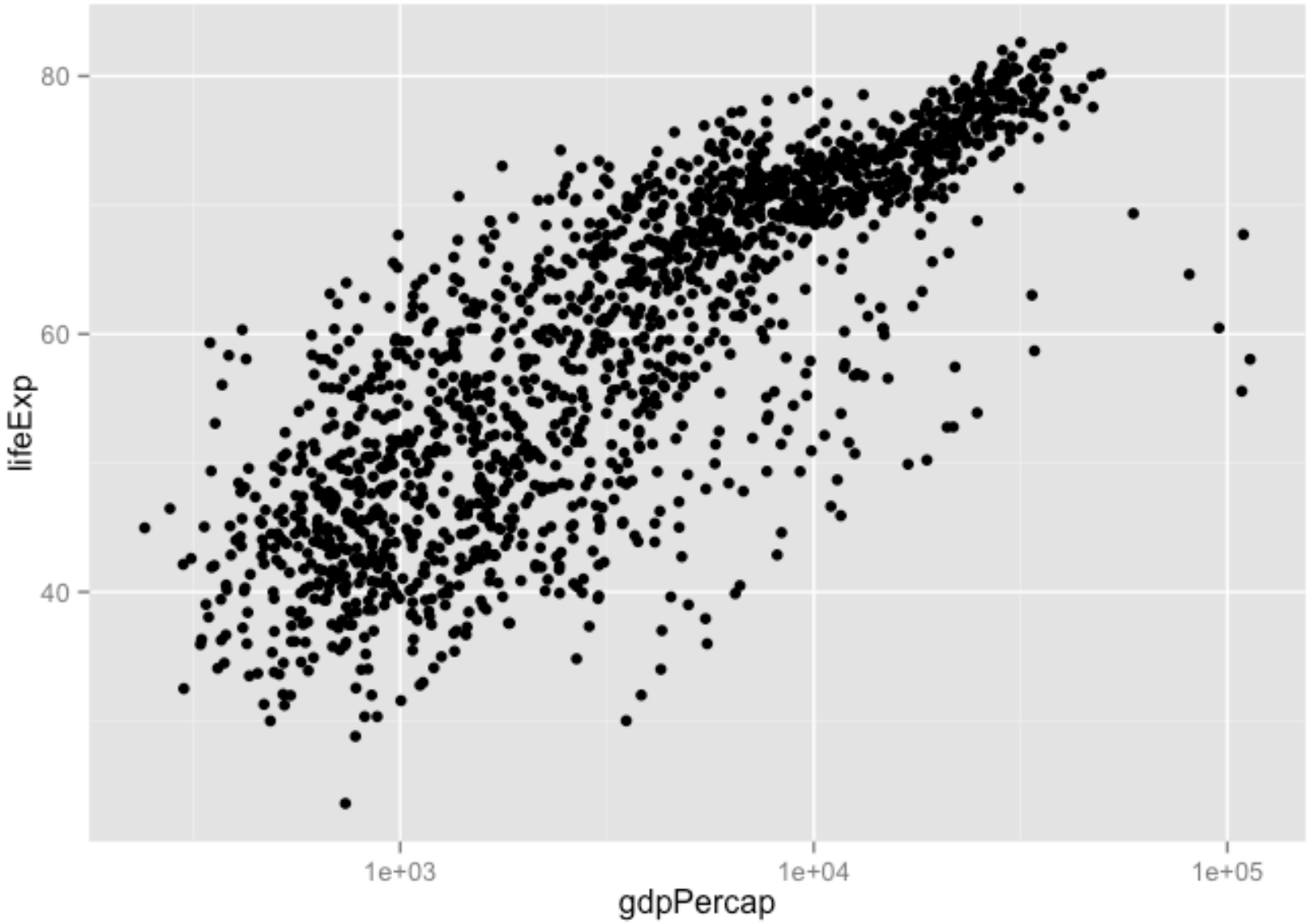
**facet**: juxtapose related mini-plots of data subsets




```
p <- ggplot(gapminder, aes(x = gdpPercap, y = lifeExp))
p + geom_point()
```

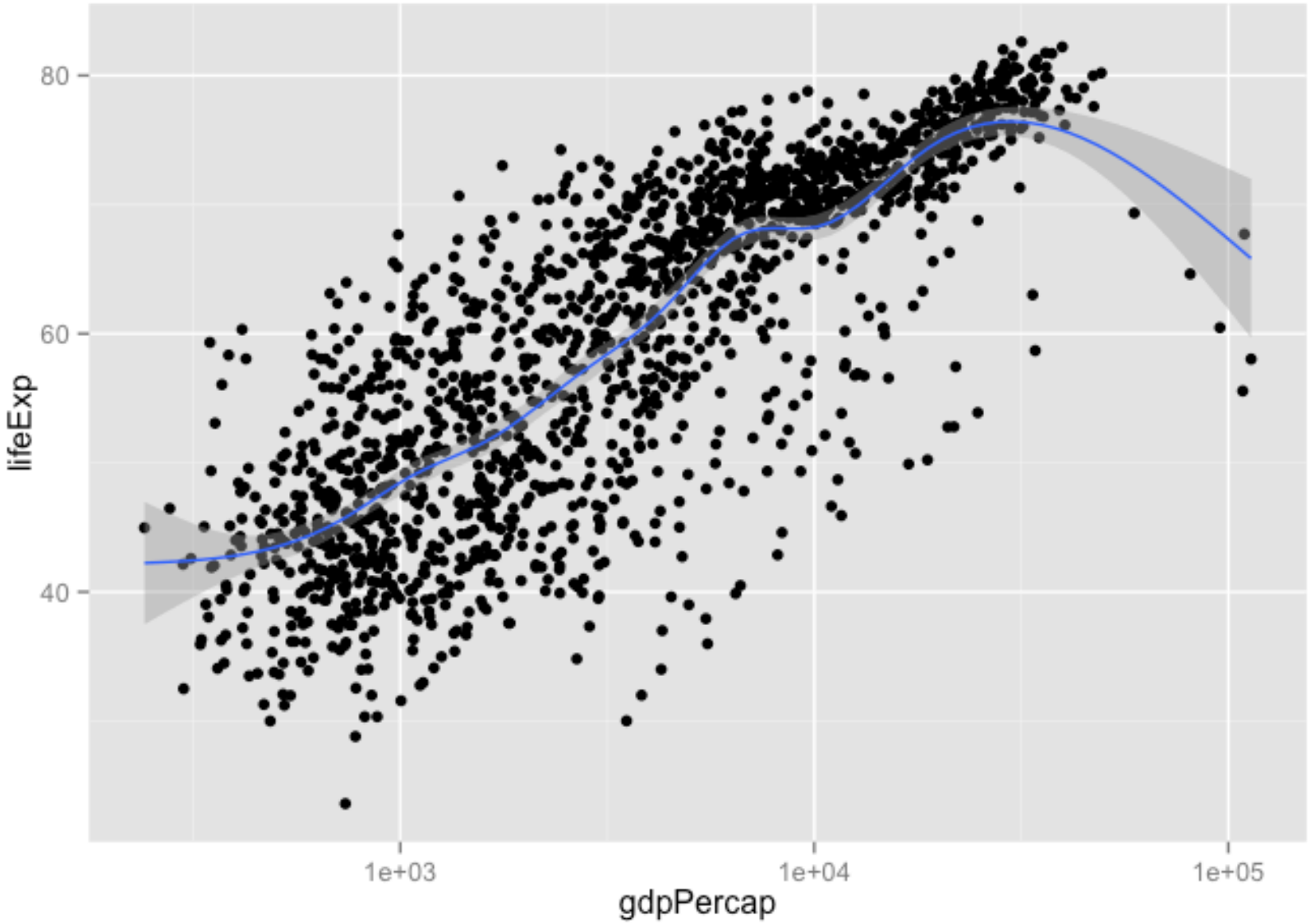



```
ggplot(gapminder, aes(x = log10(gdpPercap), y = lifeExp)) +
geom_point()
```

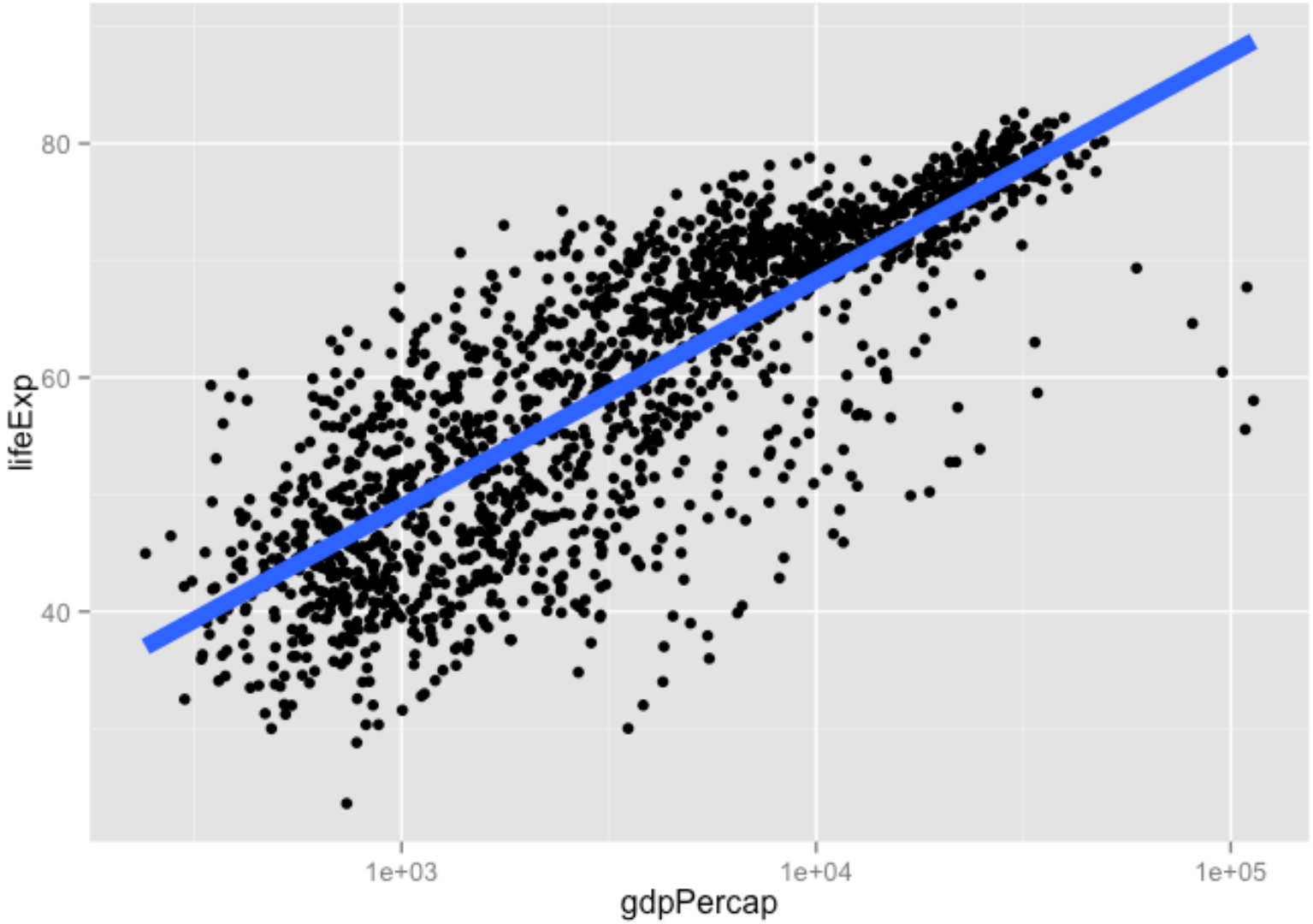



```
p + geom_point() + scale_x_log10()
```


```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, color =
continent)) +
  geom_point() + scale_x_log10()
```

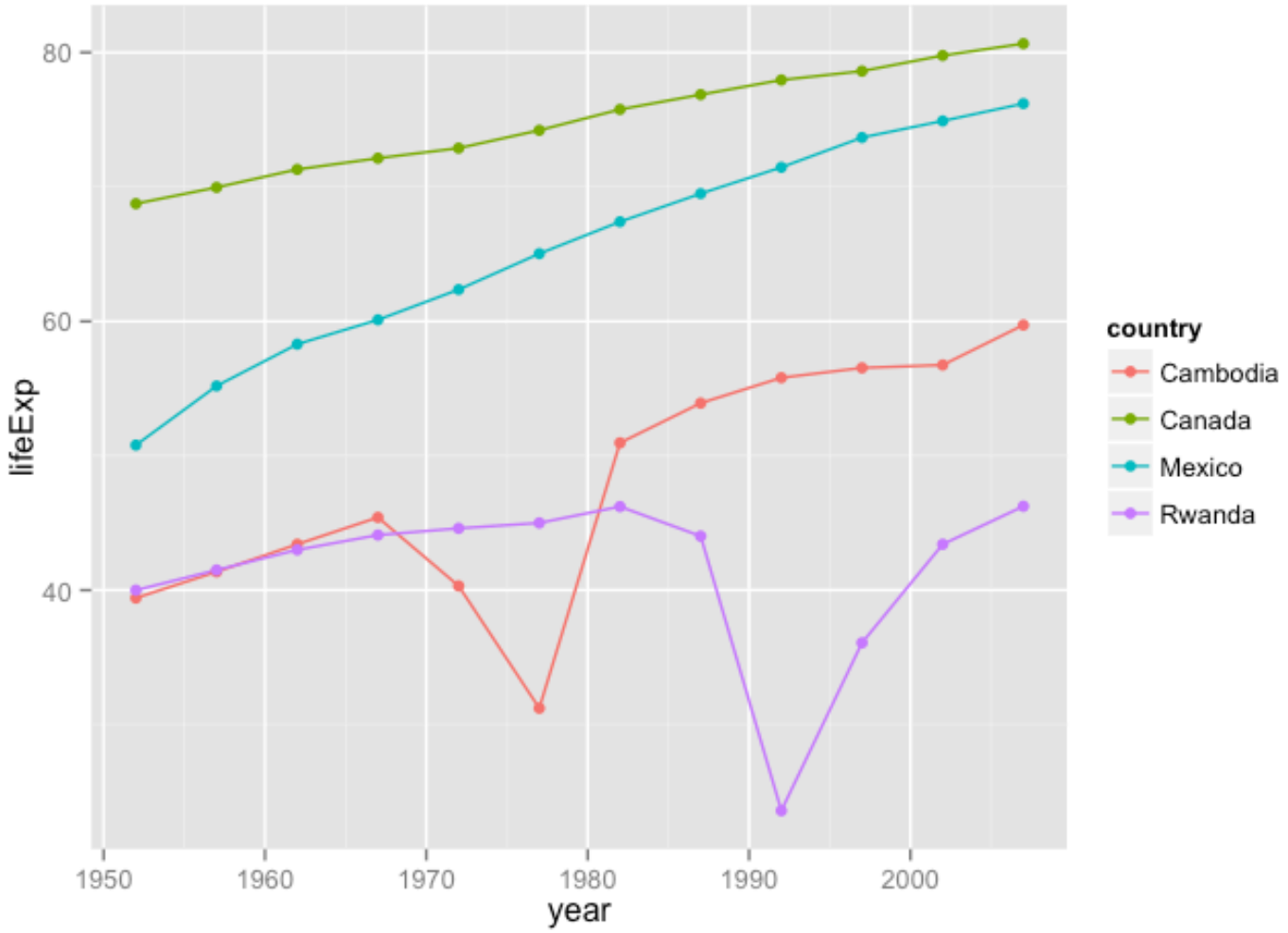



```
p + geom_point() + geom_smooth()
```

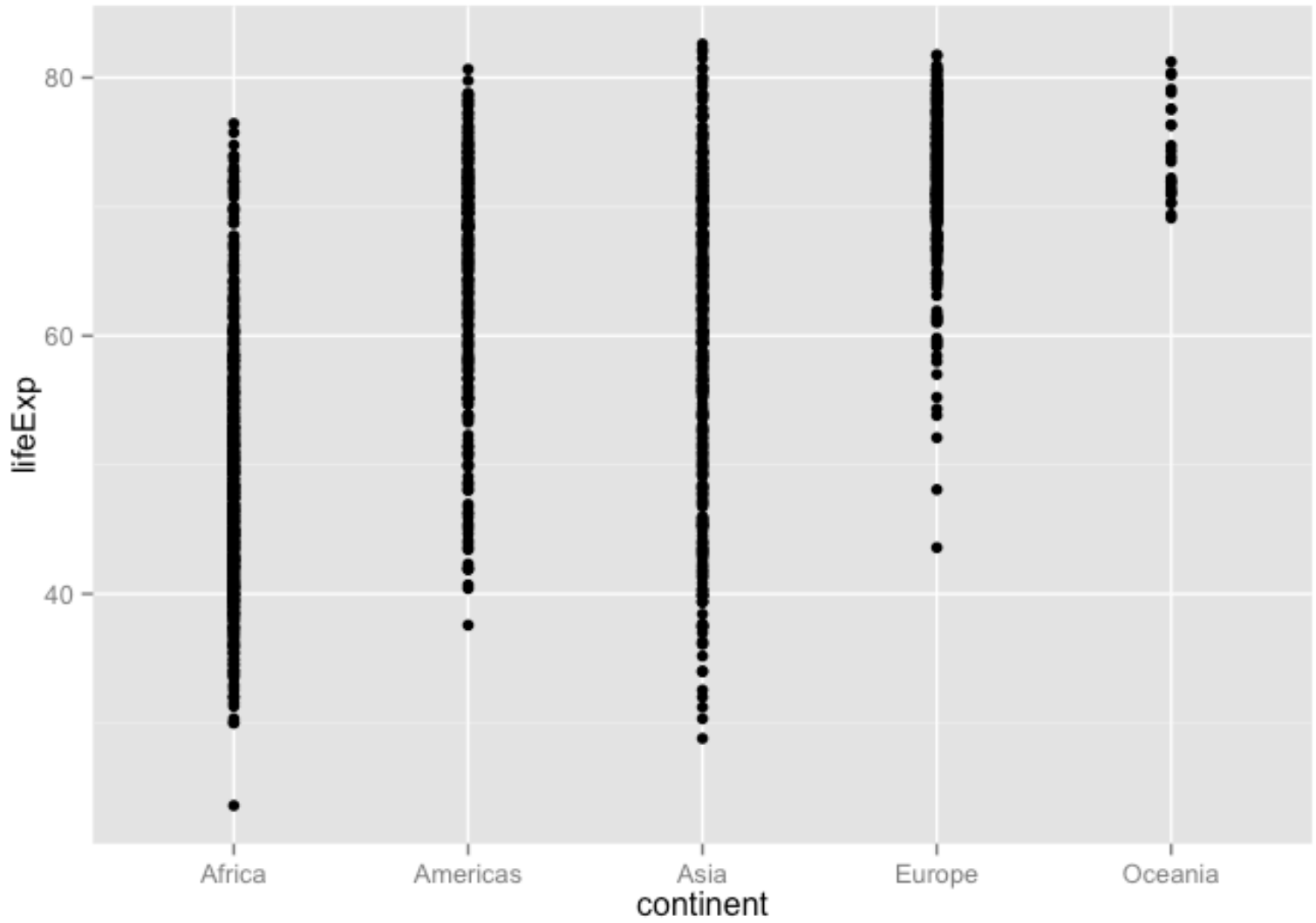



```
p + geom_point() + geom_smooth(lwd = 3, se = FALSE, method
= "lm")
```

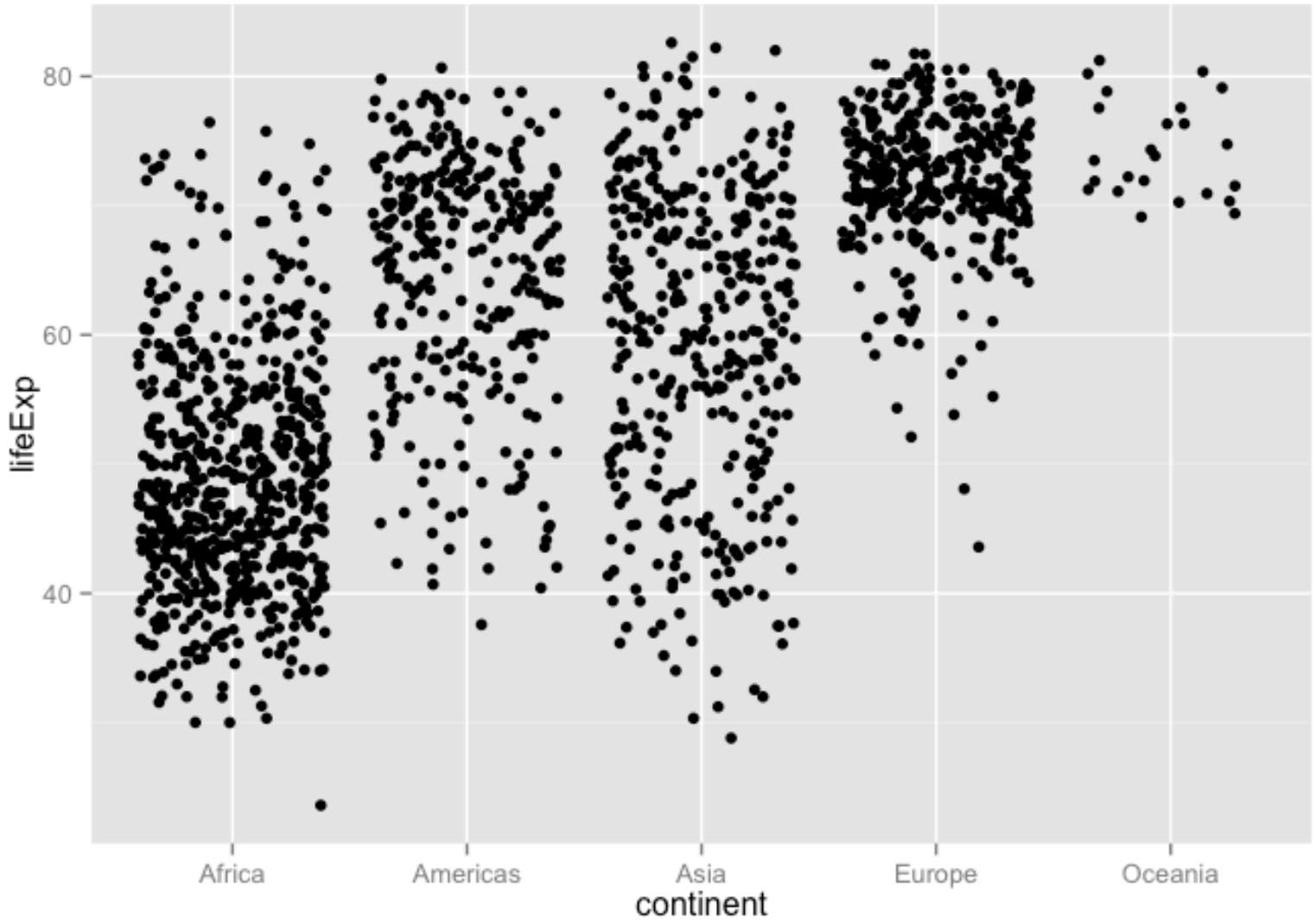

```
ggplot(subset(gapminder, country == "Zimbabwe"),
       aes(x = year, y = lifeExp)) + geom_line() + geom_point()
```



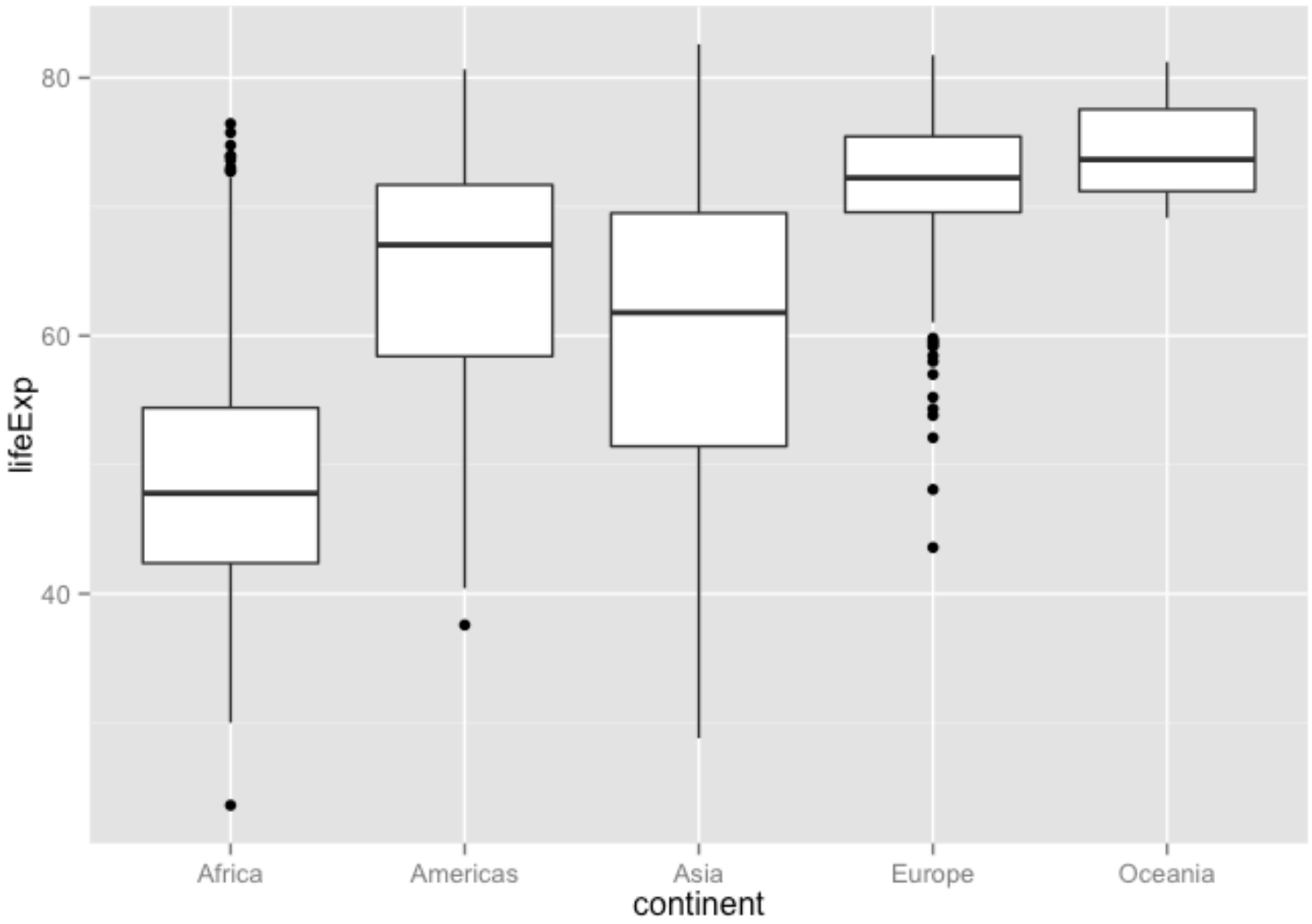

```
jCountries <- c("Canada", "Rwanda", "Cambodia", "Mexico")
ggplot(subset(gapminder, country %in% jCountries),
       aes(x = year, y = lifeExp, color = country)) +
geom_line() + geom_point()
```



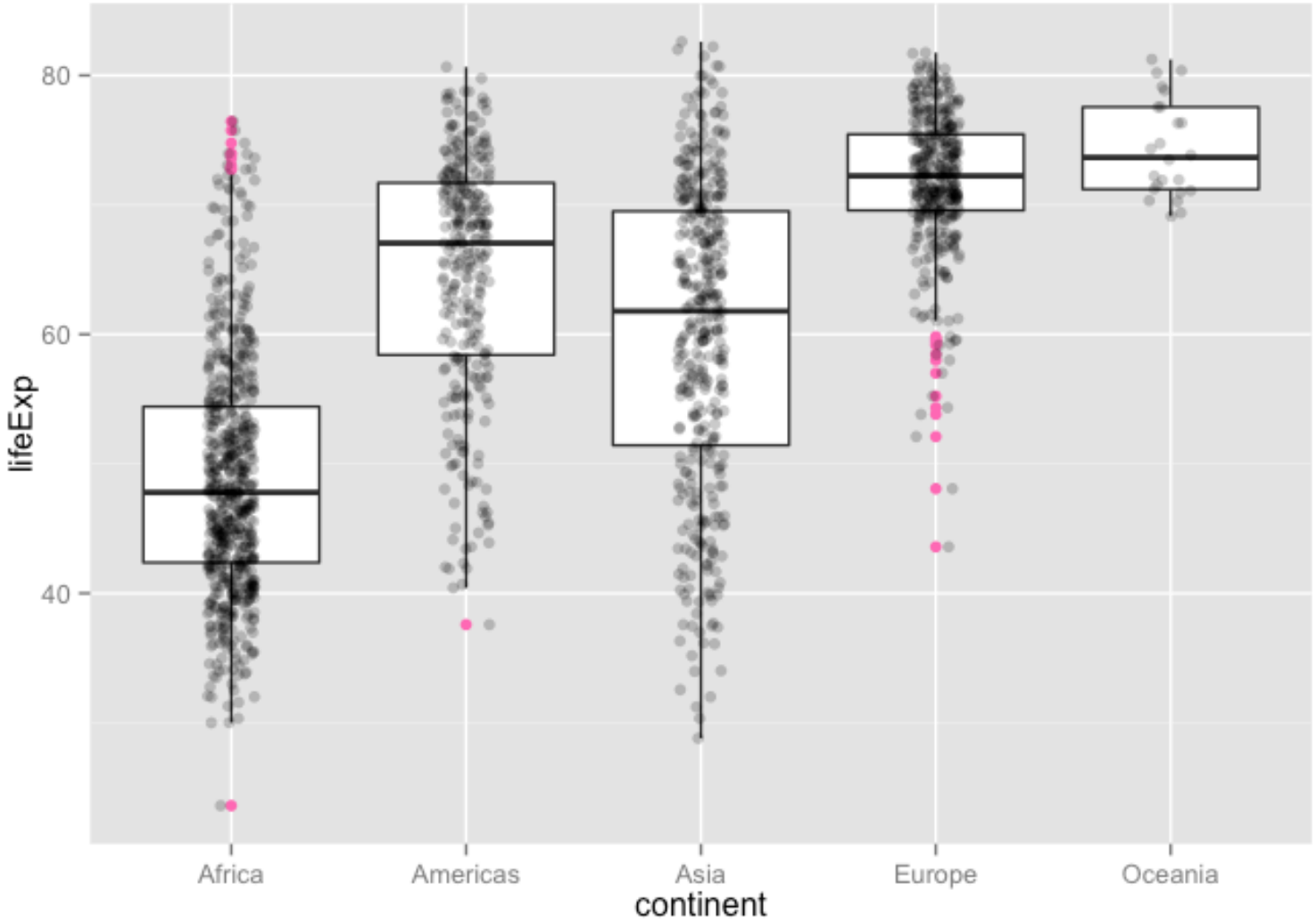

```
ggplot(gapminder, aes(x = continent, y = lifeExp)) +
geom_point()
```



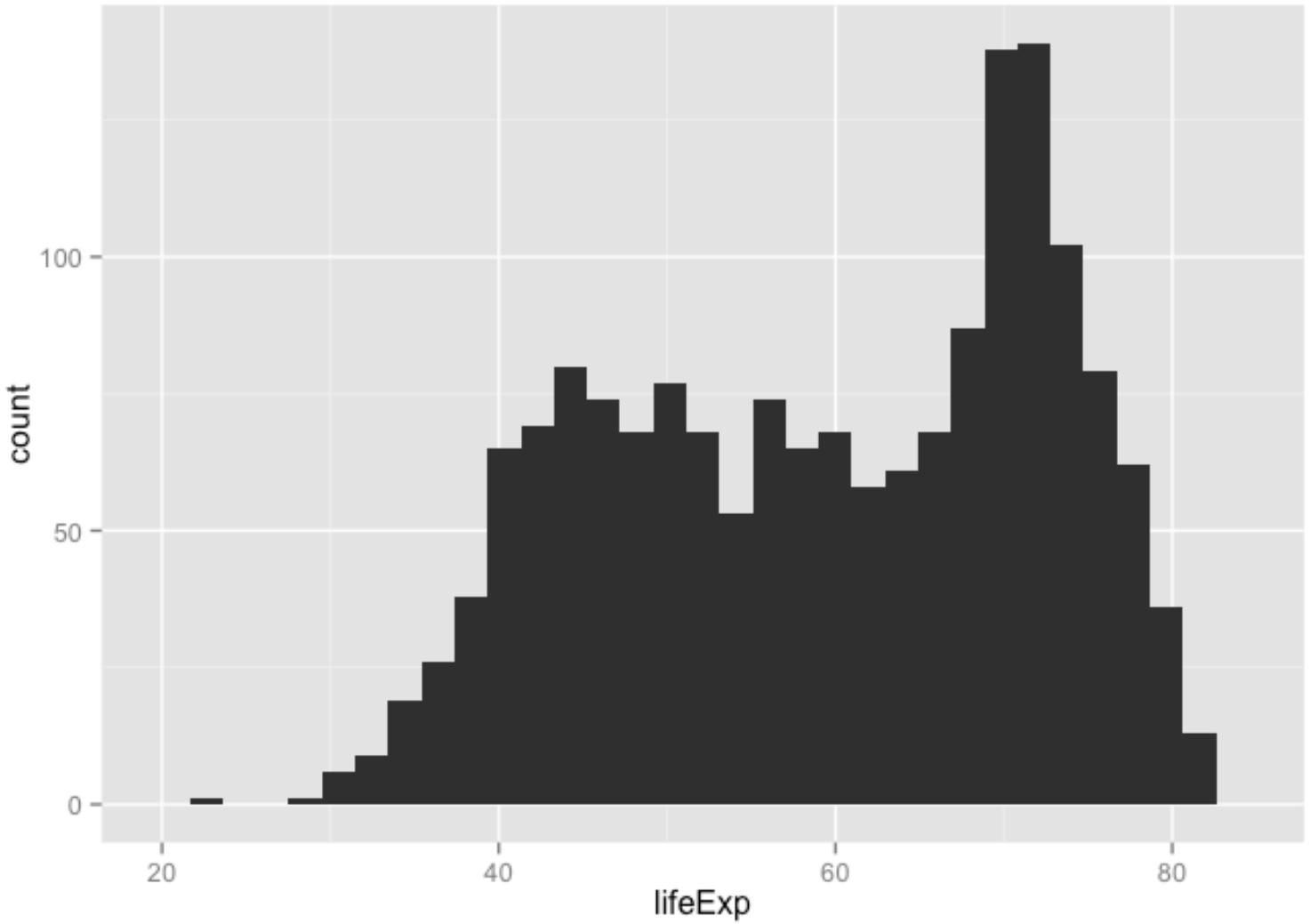

```
ggplot(gapminder, aes(x = continent, y = lifeExp)) +
geom_jitter()
```



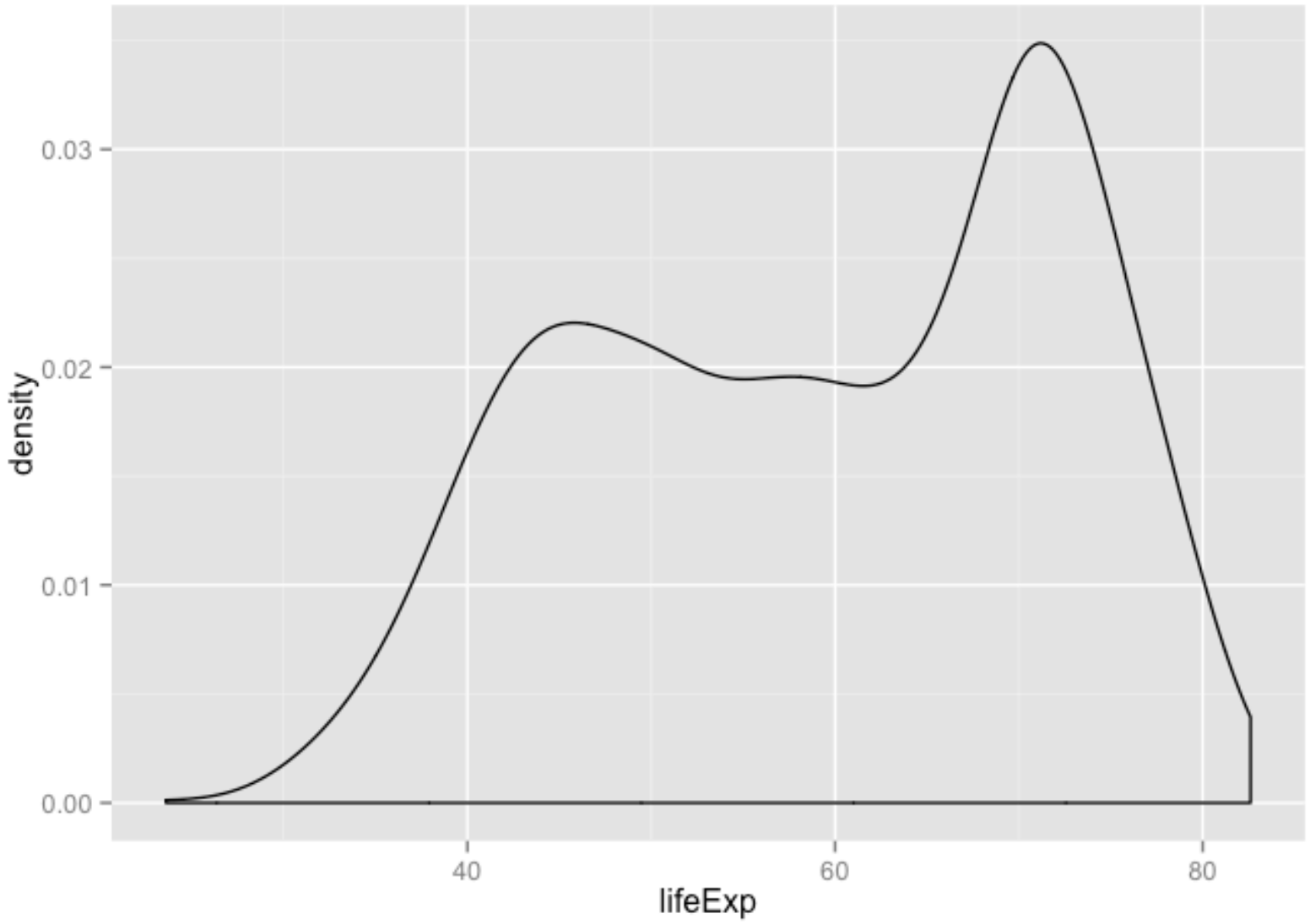

```
ggplot(gapminder, aes(x = continent, y = lifeExp)) +
geom_boxplot()
```



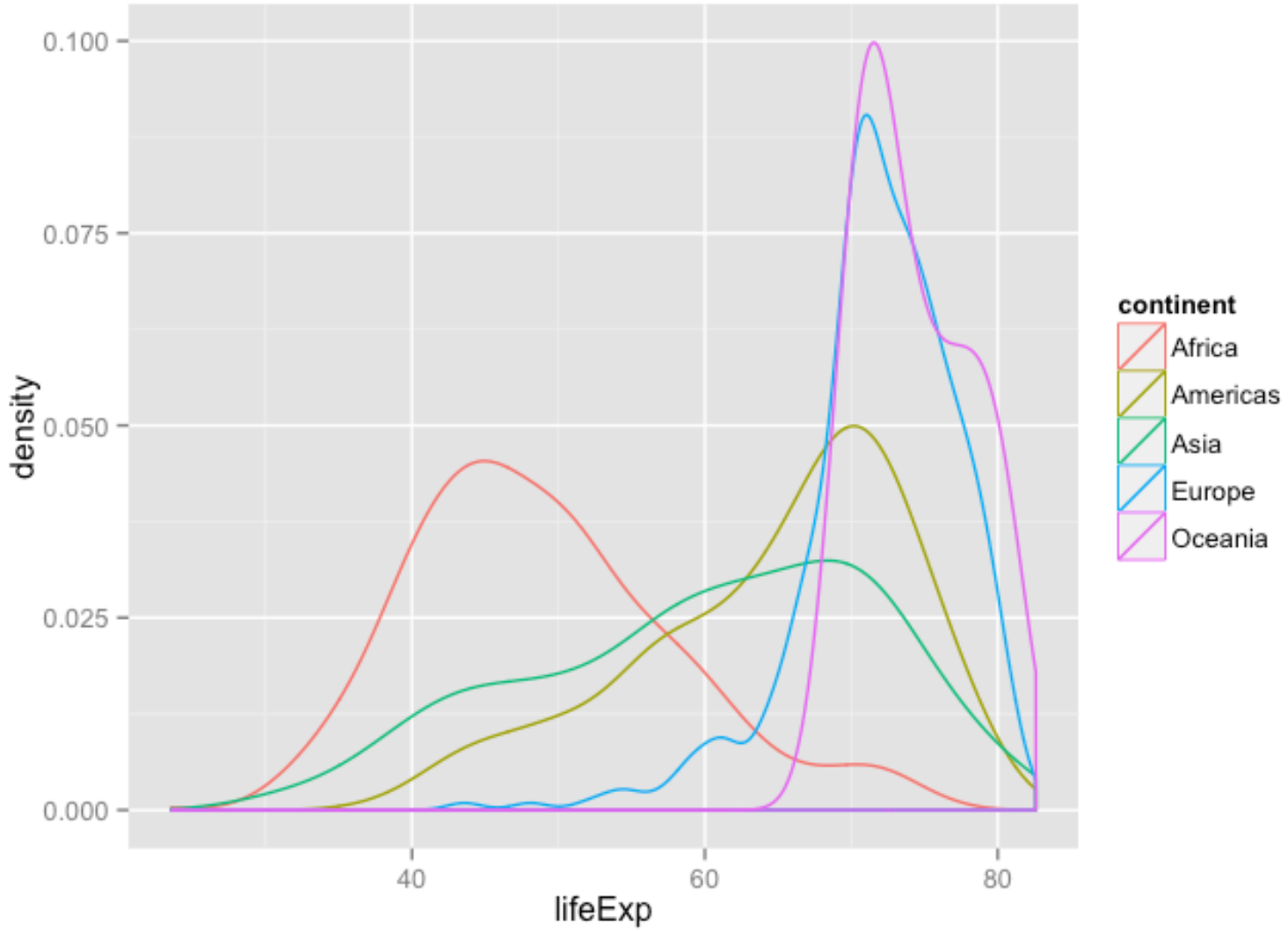

```
ggplot(gapminder, aes(x = continent, y = lifeExp)) +
  geom_boxplot(outlier.colour = "hotpink") +
  geom_jitter(position = position_jitter(width = 0.1, height =
0), alpha = 1/4)
```



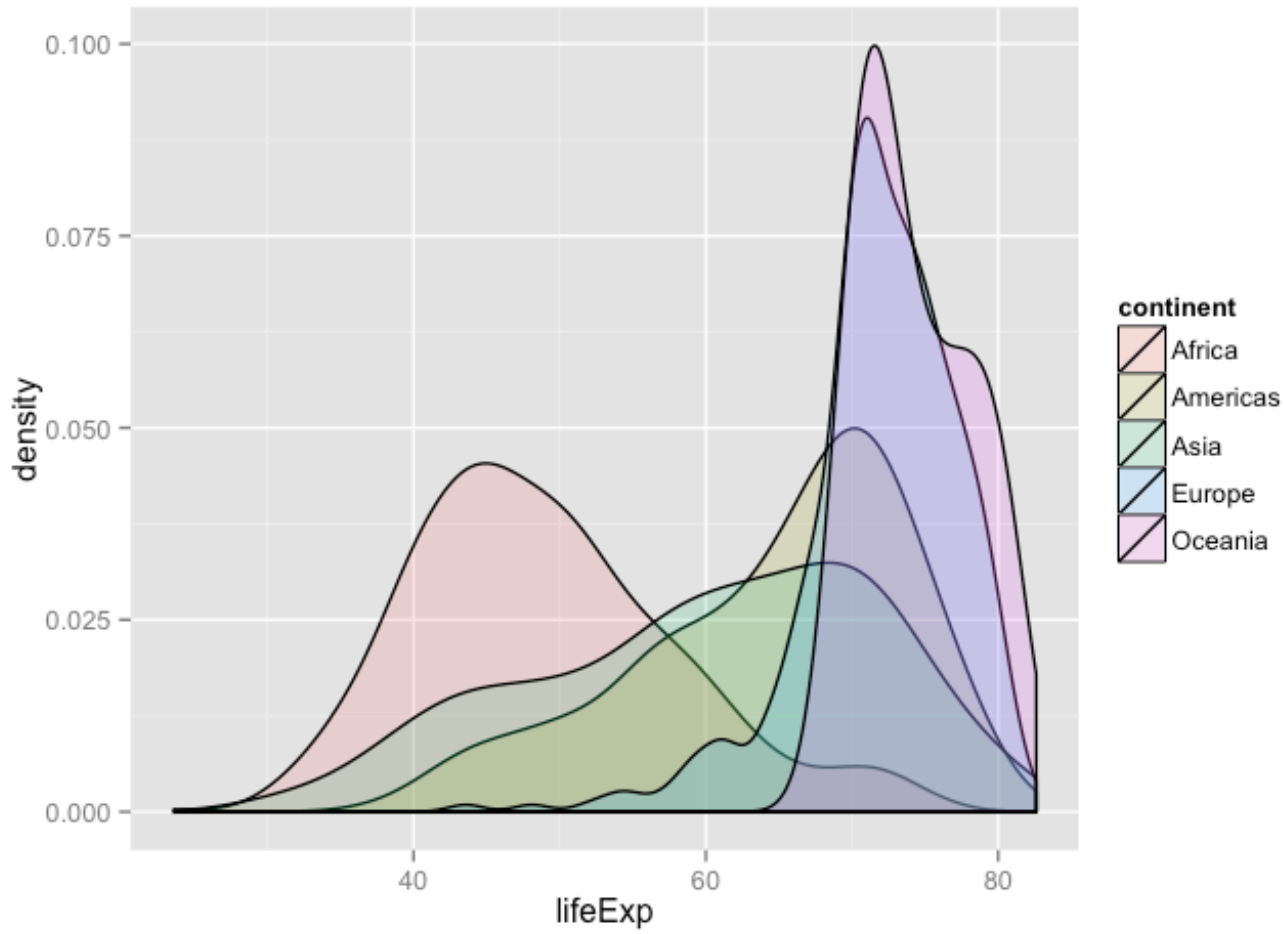

```
ggplot(gapminder, aes(x = lifeExp)) + geom_histogram()
```



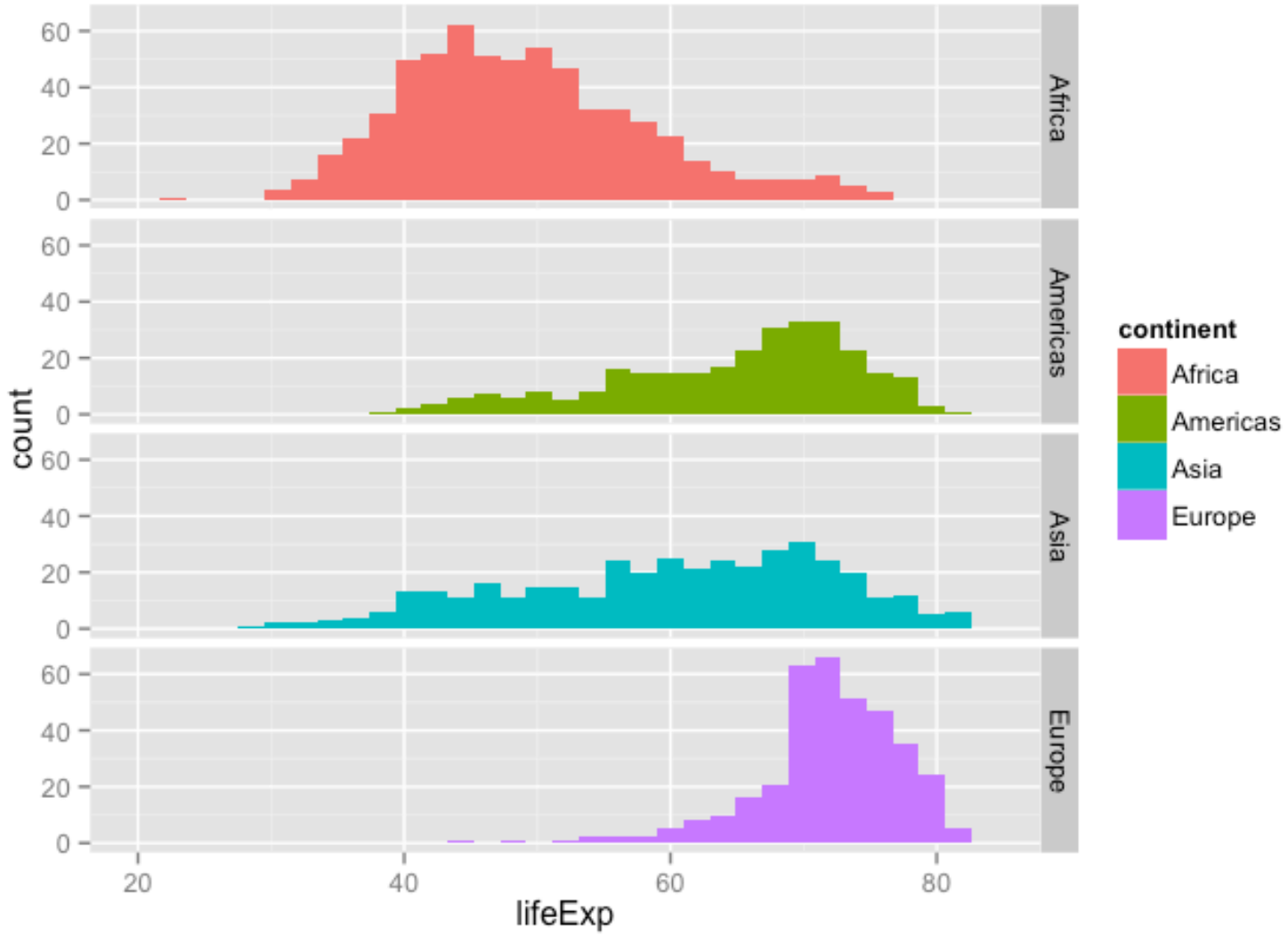

```
ggplot(gapminder, aes(x = lifeExp)) + geom_density()
```



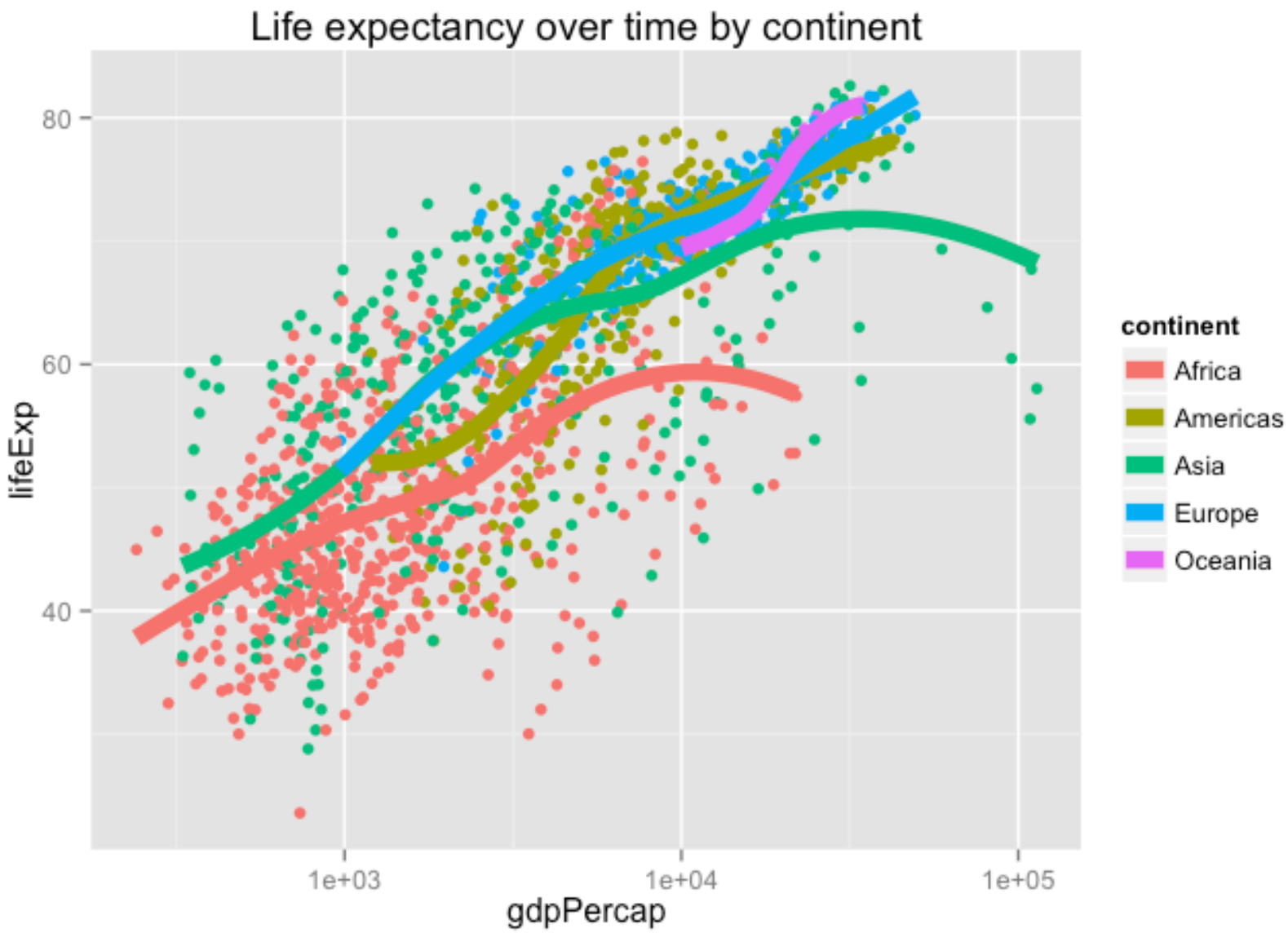

```
ggplot(gapminder, aes(x = lifeExp, color = continent)) +
geom_density()
```



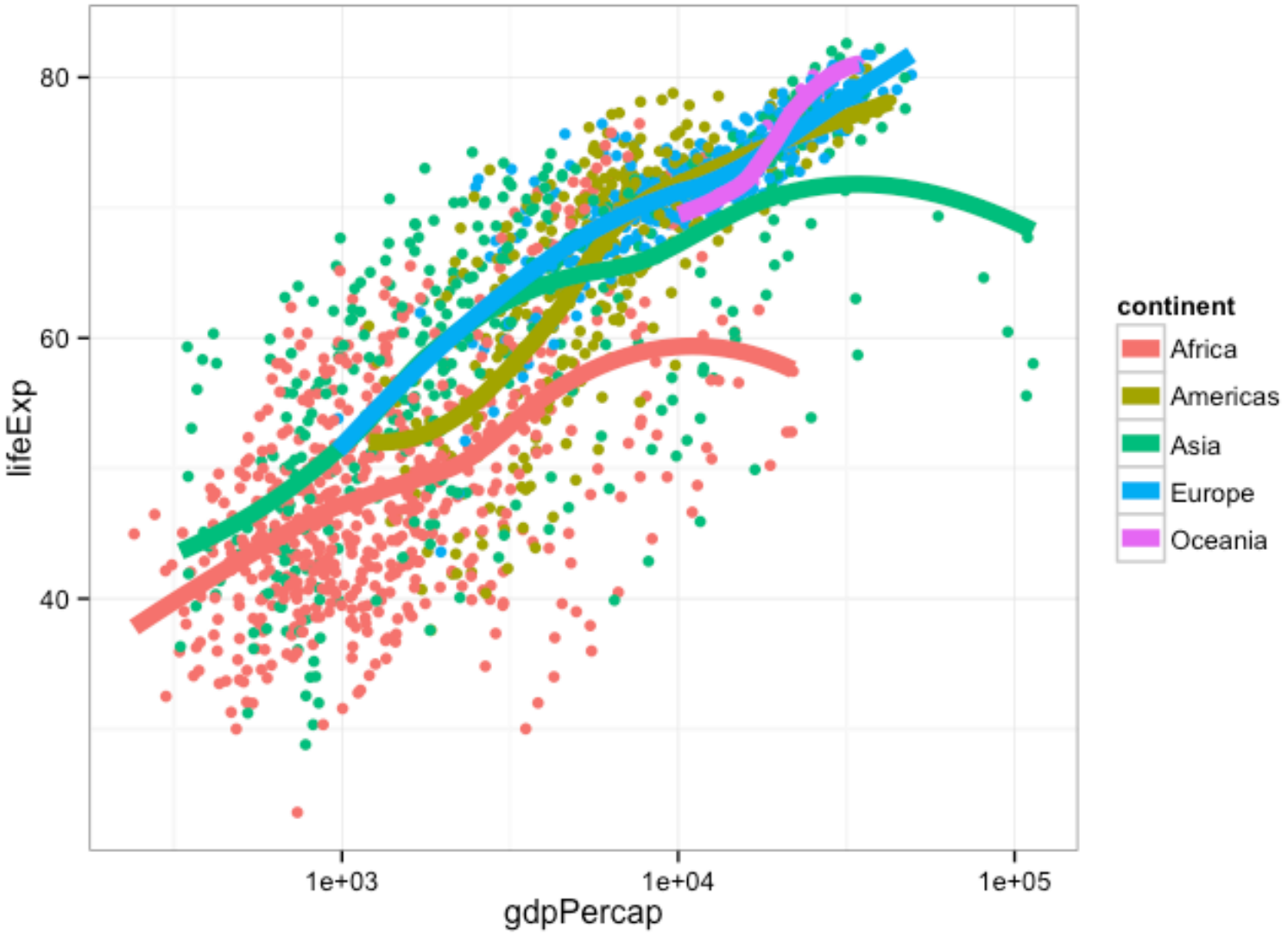

```
ggplot(gapminder, aes(x = lifeExp, fill = continent)) +
  geom_density(alpha = 0.2)
```




```
ggplot(subset(gapminder, continent != "Oceania"),
       aes(x = lifeExp, fill = continent)) +
geom_histogram() +
  facet_grid(continent ~ .)
```




```
p + ggtitle("Life expectancy over time by continent")
```


80
60
40
lifeExp
1e+03
1e+04
1e+05
gdpPercap
continent
Africa
Americas
Asia
Europe
Oceania

```
p + theme_bw()
```

# QUIZ TIME

?

Постройте график зависимоти продолжительности жизни от ВВП на душу населения (GDP per capita) для выборки gapminder. Сделайте логарифмическое шкалирование по оси GDP. Раскрасьте точки по континентам и постройте линии тренда с помощью линейной модели.

Выберите верные утверждения:
a. линия тренда для Азии лежит выше линии тренда для Африки
b. линия тренда для Европы лежит выше линии тренда для Азии
c. линия тренда для Африки лежит выше линии тренда для Азии
d. линия тренда для Америк лежит выше линии тренда для Азии