Основы программирования в R

Визуализация с ggplot2: часть 1

Алла Тамбовцева, НИУ ВШЭ

Содержание

Введение в ggplot2

Библиотека ggplot2 устанавливается вместе с tidyverse и позволяет строить красивые графики. Обратимся к ней:

```
library(tidyverse)
```

Чтобы разобраться с логикой построения графиков с помощью ggplot2, пока будем работать с простыми данными — данными по температуре тела бобров beaver1 (к политологии вернемся на следующем занятии). Подготовим датафрейм:

```
beav <- beaver1 # загрузим базу - она встроена в R
beav$id <- 1:nrow(beaver1) # добавим id
```

Теперь перейдем к ggplot2. Можно считать, что у библиотеки ggplot2 есть своя философия, поняв которую, строить графики гораздо легче.

Во-первых, графики с ggplot2 многослойные, то есть строятся они поэтапно, по слоям. Сначала указывается датафрейм, с которым мы работаем, и интересующие нас показатели (первый слой), затем указывается тип графика (второй слой), затем настройки для подписей, легенды и прочее (остальные слои). Все слои добавляются через +.

Во-вторых, для любого графика указывается функция aes(), сокращенно от *aesthetics*, в качестве аргументов которой задаются переменные интереса (которые хотим отобразить на графике), а также элементы оформления графика, которое непосредственно связано с переменными в датафрейме. О чем речь? Проще понять на примерах.

Пример 1. Строим диаграмму рассеяния для роста и веса человека, хотим, чтобы все точки на диаграмме рассеяния были зелёными.

Пример 2. Строим диаграмму рассеяния для роста и веса человека, хотим, чтобы точки на диаграмме рассеяния, соответствующие женщинам, были красными, а мужчинам — синими.

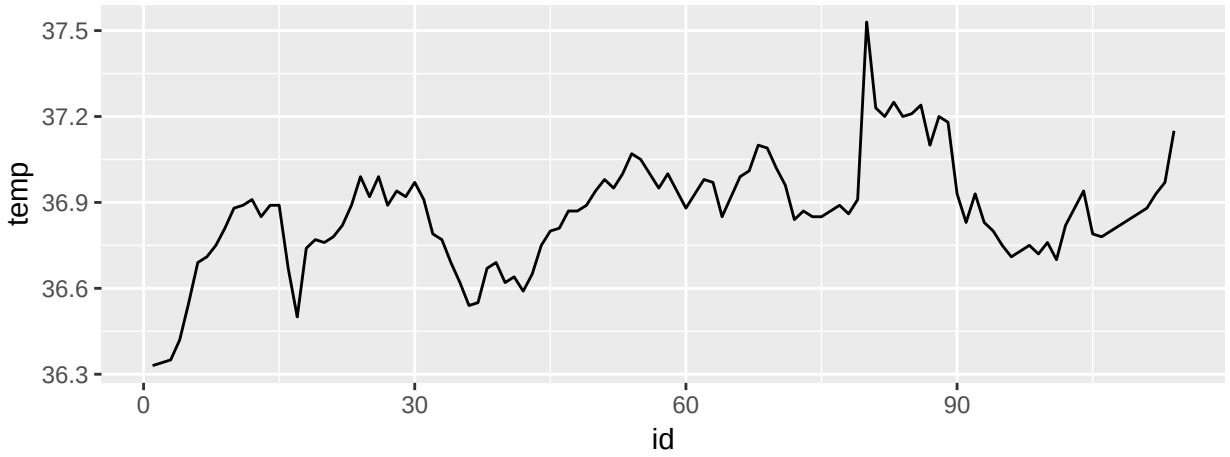
В первом примере оформление графика никак не связано со значениями переменных в датафрейме, все точки закрашиваем одним цветом. Во втором примере цвет точек зависит от значения переменной пол, то есть оформление графика связано с переменными в датафрейме. Как увидим позже, в случаях, аналогичным первому, цвет точек будет определяться за пределами aes(), второму — внутри aes().

Линейные графики (*line plots*)

Построим первый график. До этого занятия мы не обсуждали линейные графики (*line plots*), но все с ними так или иначе сталкивались, когда следили за динамикой каких-то количественных показате-

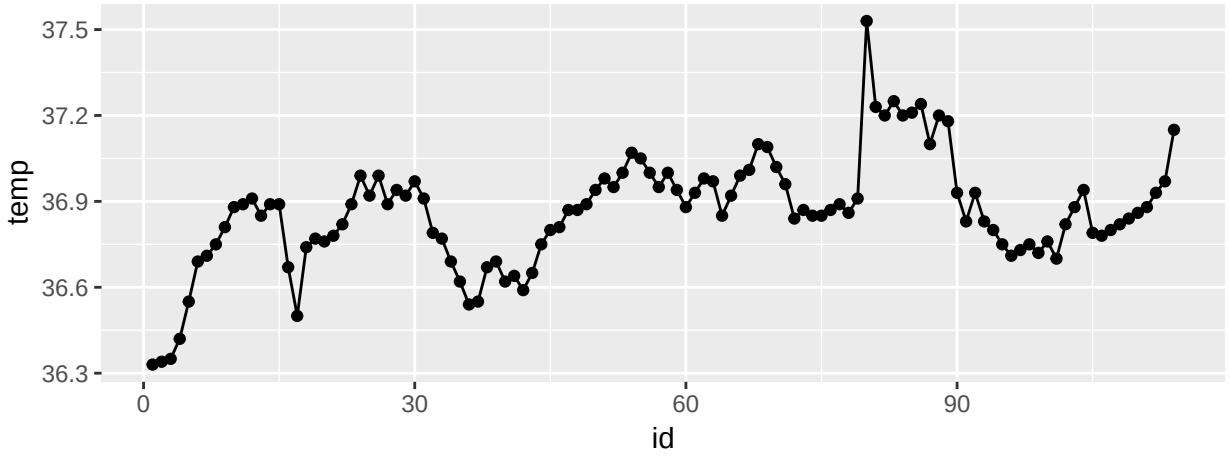
лей. Попробуем визуализировать динамику температуры тела бобров в течении времени (в качестве показателя времени будем использовать `id` замера температуры, так как все замеры производились последовательно, с интервалом в 10 минут).

```
# в aes - показатели по оси x и y
# через + указан тип графика geom_line()

ggplot(data = beav, aes(x = id, y = temp)) + geom_line()
```



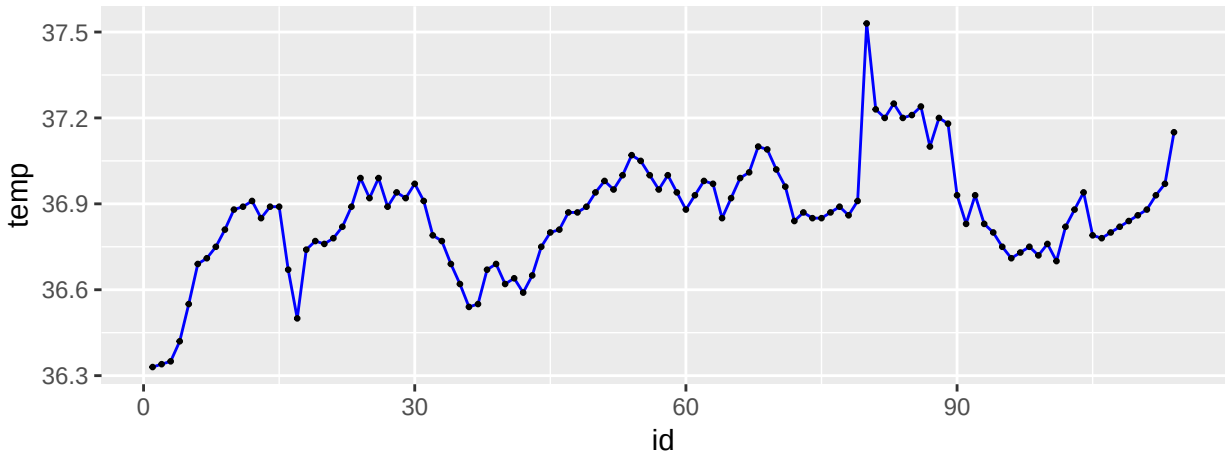
Типы графиков можно сочетать. Добавим точки (чтобы получились точки, соединенные линиями):

```
# два типа: geom_line и geom_point

ggplot(data = beav, aes(x = id, y = temp)) +
  geom_line() +
  geom_point()
```



Цвета и типы точек и линий можно изменять. Сделаем это!

```
# синие линии
# точки поменьше

ggplot(data = beav, aes(x = id, y = temp)) +
  geom_line(color = "blue") +
  geom_point(size = 0.5)
```

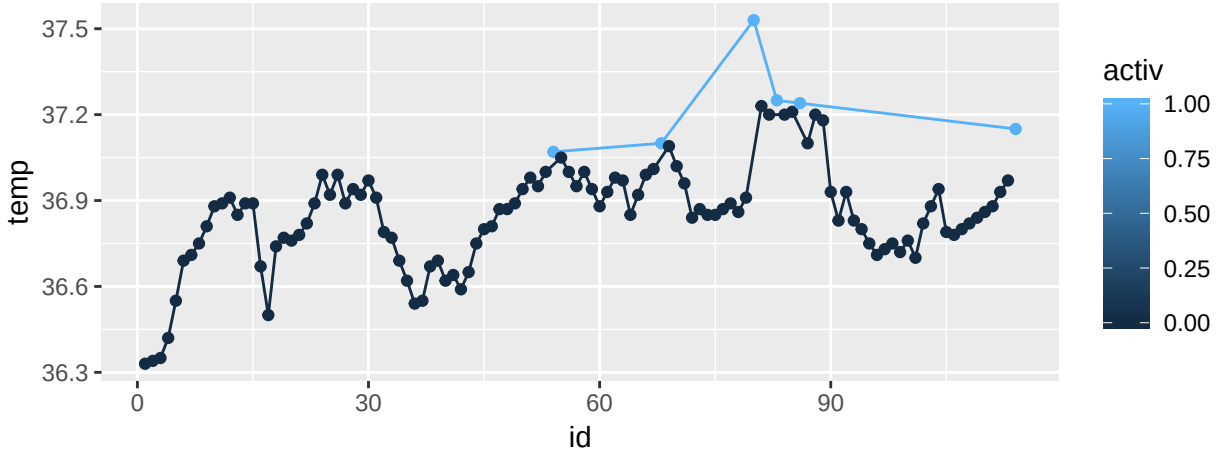


Разных опций, конечно, много. Чтобы узнать о всех возможностях, можно запросить `help` отдельно для оформления точек или линий:

```
?geom_point
?geom_line
```

Посмотрим теперь, в каких случаях параметры оформления графика имеет смысл указывать внутри `aes()`. В «бобрином» датафрейме у нас есть переменная `activ` — активность бобров (0 — не активен, 1 — активен). Представим, что мы хотим построить два линейных графика в одной плоскости: один для неактивных бобров, другой — для активных.

```
# group - группировка по переменной, чтобы получилось 2 отдельных графика
# color - чтобы разные группы точек были разного цвета (в зависимости от значений activ)

ggplot(data = beav, aes(x = id, y = temp,
                        group = activ,
                        color = activ)) +
  geom_line() + geom_point()
```

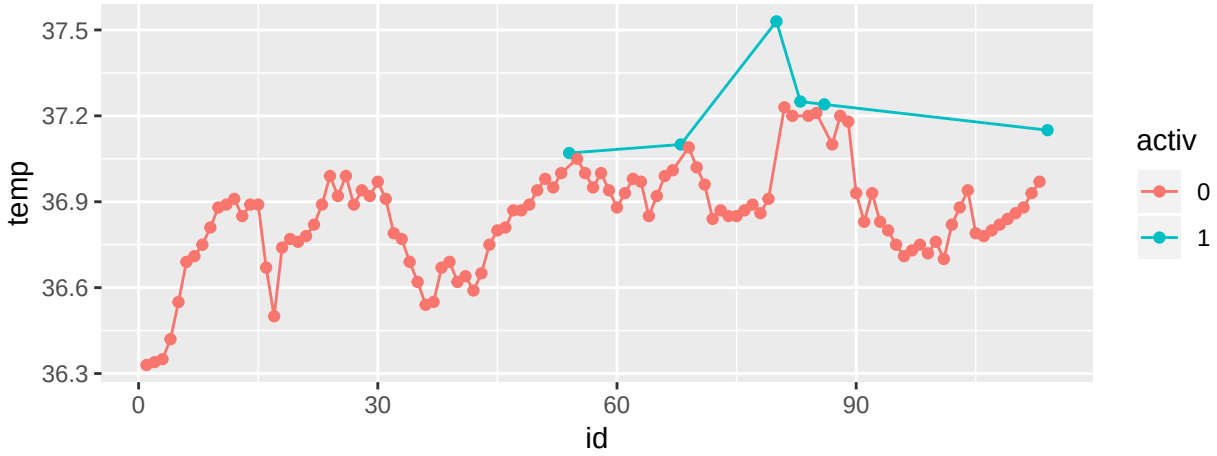


Внимание, вопрос: почему график правильный, а легенда у него такая странная? Переменная `activ` принимает всего два значения, 0 и 1, а тут целая шкала от 0 до 1 образовалась... Эта проблема возникла потому, что у нас в базе данных переменная `activ` не факторная (тип *factor*), а числовая (тип *numeric*). Чтобы получить правильную легенду, скорректируем тип переменной:

```
beav <- beav %>% mutate(activ = factor(activ))
```

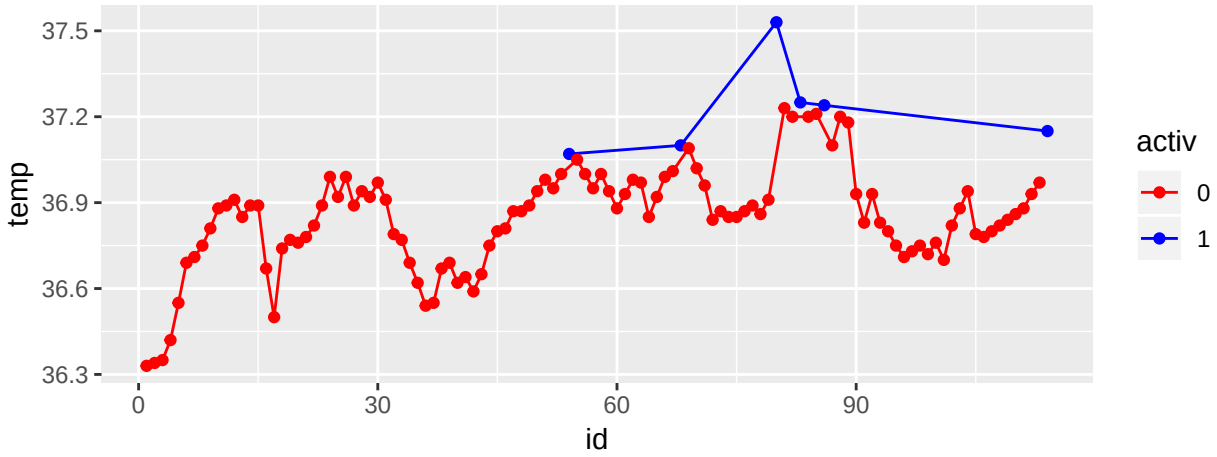
Посмотрим теперь:

```
ggplot(data = beav,
       aes(x = id, y = temp,
           group = activ,
           color = activ)) +
  geom_line() +
  geom_point()
```



Теперь все верно. Но цвета поменялись. По умолчанию в R разбивка на две группы —разбивка по признаку «пол», поэтому цвета получились такими. Конечно, их можно поменять:

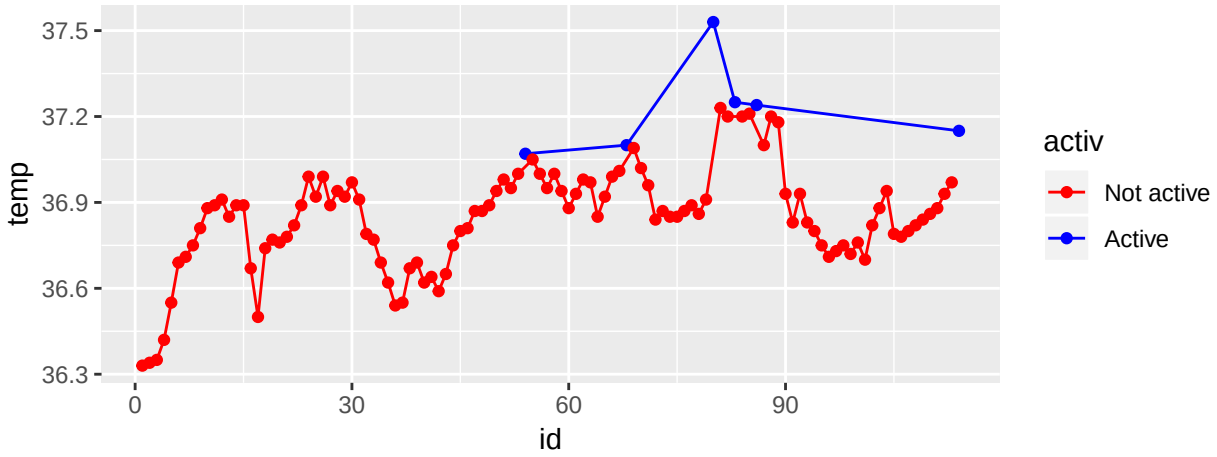
```
# scale_color_manual - задаем вектор значений цветов вручную

ggplot(data = beav,
       aes(x = id, y = temp,
           group = activ,
           color = activ)) +
  geom_line() +
  geom_point() +
  scale_color_manual(values = c("red", "blue"))
```



А теперь с помощью этого же слоя `scale_color_manual` поменяем названия групп, указанных в легенде графика:

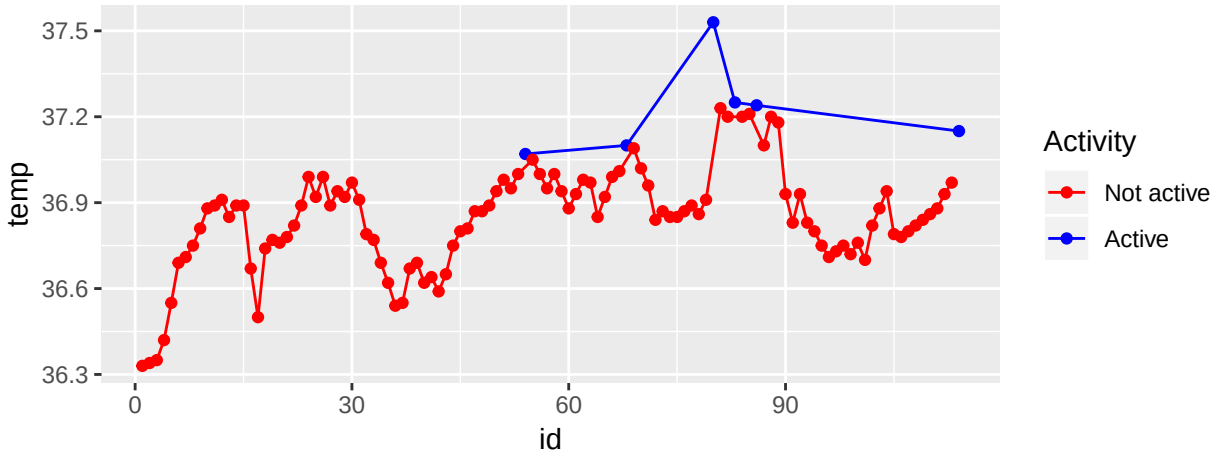
```
# то же + аргумент labels

ggplot(data = beav,
```

```
      aes(x = id, y = temp,
          group = activ,
          color = activ)) +
  geom_line() +
  geom_point() +
  scale_color_manual(values = c("red", "blue"),
  labels = c("Not active", "Active"))
```



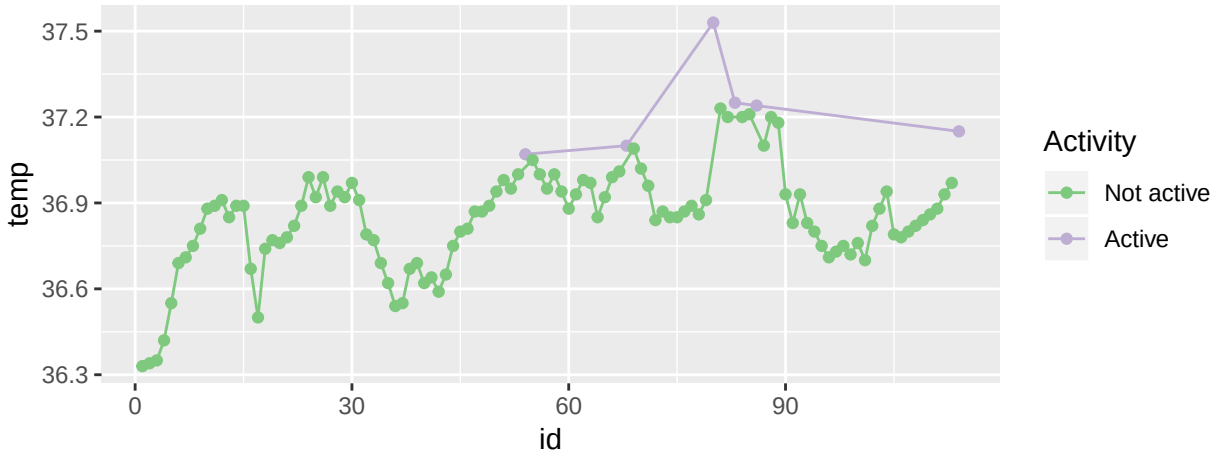
И поменяем заголовок в легенде:

```
# то же + аргумент name

ggplot(data = beav,
       aes(x = id, y = temp,
           group = activ,
           color = activ)) +
  geom_line() +
  geom_point() +
  scale_color_manual(values = c("red", "blue"),
  labels = c("Not active", "Active"),
  name = "Activity")
```



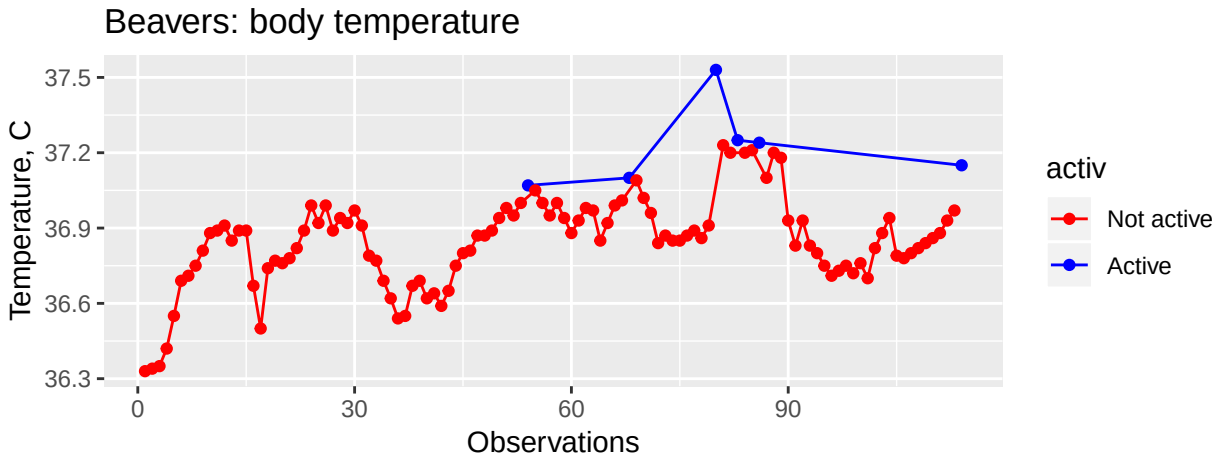
Функций вида `scale_color_` существует несколько. Например, вместо того, чтобы задавать цвета самостоятельно, можно взять слой `scale_color_brewer` и выбрать одну из встроенных палитр (как в Python):

```
# палитра Accent

ggplot(data = beav,
       aes(x = id, y = temp,
           group = activ,
           color = activ)) +
  geom_line() +
  geom_point() +
  scale_color_brewer(palette = "Accent",
                     labels = c("Not active", "Active"),
                     name = "Activity")
```



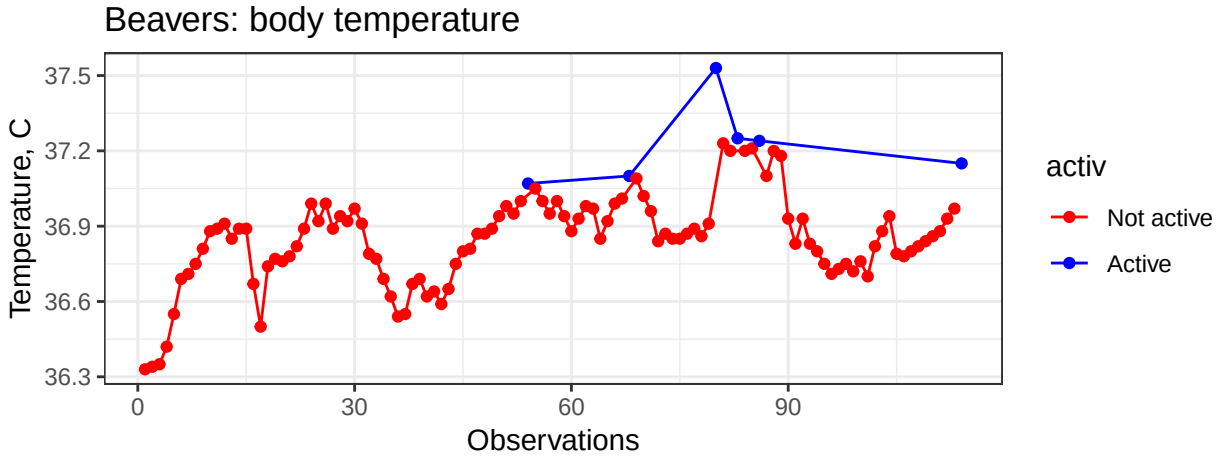
Теперь осталось узнать, как подписывать оси на графике и добавлять заголовок. Для всего этого есть один слой `labs` (можно и иначе, есть отдельные слои для заголовков, подзаголовков и прочих подписей):

```
# title - заголовок
# x - подпись оси x
# y - подпись оси y

ggplot(data = beav, aes(x = id, y = temp,
                        group = activ,
                        color = activ)) +
  geom_line() + geom_point() +
  scale_color_manual(values = c("red", "blue"),
                     labels = c("Not active", "Active")) +
  labs(title = "Beavers: body temperature",
       x = "Observations",
       y = "Temperature, C")
```



В завершение нашего первого знакомства с `ggplot2` поменяем тему графика (`theme`). По умолчанию график строится на сером фоне, но фон можно сделать, например, белым:

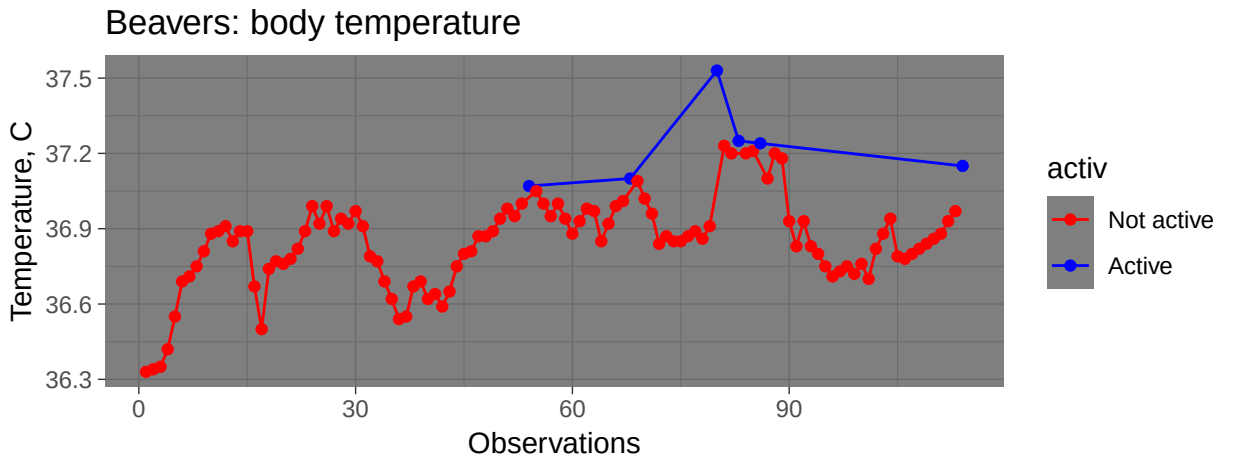
```
# добавляем еще слой с theme
# bw: black-white

ggplot(data = beav, aes(x = id, y = temp,
                        group = activ,
                        color = activ)) +
  geom_line() +
  geom_point() +
  scale_color_manual(values = c("red", "blue"),
                     labels = c("Not active", "Active")) +
  labs(title = "Beavers: body temperature",
       x = "Observations",
       y = "Temperature, C") +
  theme_bw()
```



Или, наоборот, темным (правда, здесь это будет не очень удачно смотреться):

```
ggplot(data = beav, aes(x = id, y = temp,
                        group = activ,
                        color = activ)) +
  geom_line() +
  geom_point() +
  scale_color_manual(values = c("red", "blue"),
```

```
                    labels = c("Not active", "Active")) +
  labs(title = "Beavers: body temperature",
       x = "Observations",
       y = "Temperature, C") +
  theme_dark()
```



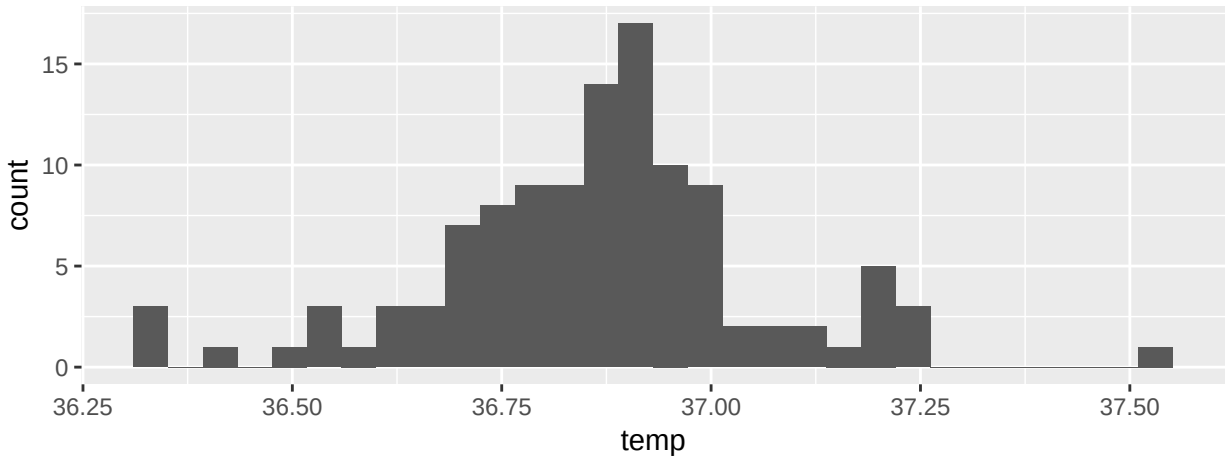
А теперь перейдем к другим графикам.

Гистограммы и сглаженные графики плотности распределения

Для чего нужны гистограммы, мы уже обсуждали. Гистограммы строятся для визуализации формы распределения количественного показателя. Построим гистограмму для температуры тела бобров:

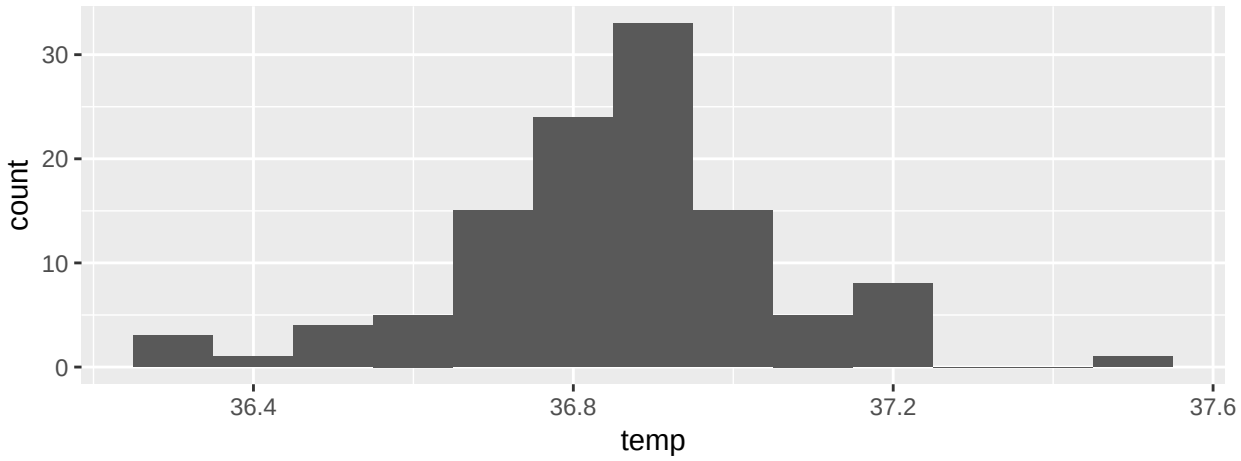
```
ggplot(data = beav, aes(x = temp)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



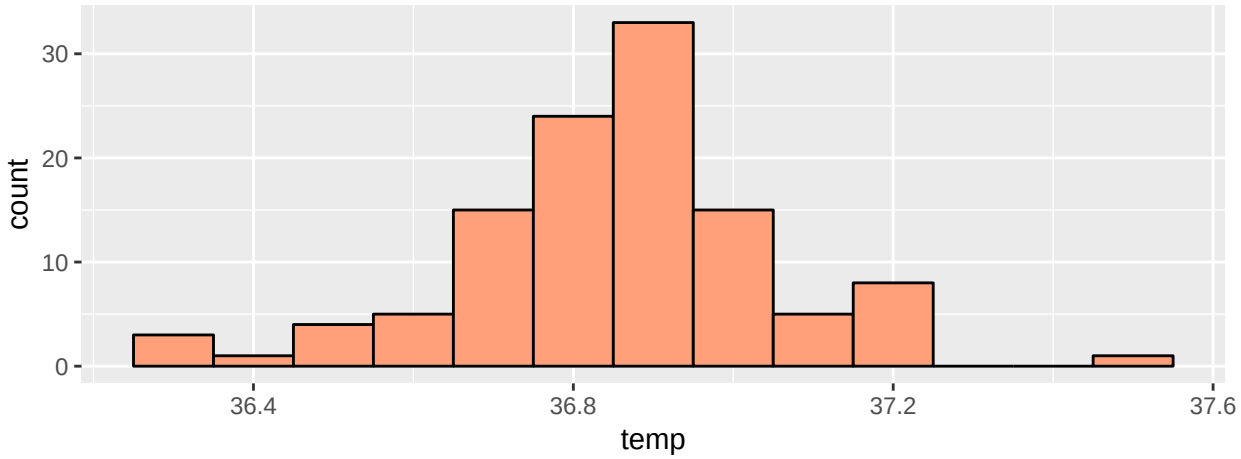
Пока выглядит несильно симпатично. Начинаем исправлять. Прежде всего, изменим шаг гистограммы, то есть ширину столбца. При построении этого графика R выдал предупреждение, что по умолчанию было построено 30 столбцов, что в данном случае может быть некорректно. Выставим шаг (`binwidth = 1`) вручную:

```
ggplot(data = beav, aes(x = temp)) +
  geom_histogram(binwidth = 0.1)
```



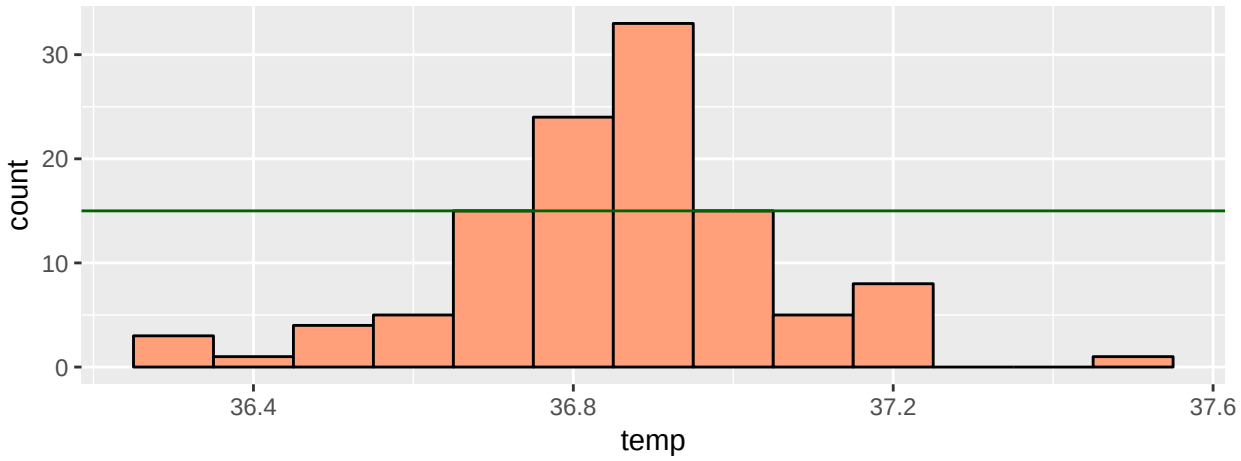
Теперь поменяем цвет. При изменении цвета «заполненных» (состоящих не из отдельных линий и точек) графиков нужно помнить, что есть два параметра: `color` и `fill`. Параметр color отвечает за цвет границ графика, а за не цвет их заливки. А уже `fill` — как раз за заливку.

```
# гистограмма цвета лосося
# столбцы которой очерчены черной линией

ggplot(data = beav, aes(x = temp)) +
  geom_histogram(binwidth = 0.1,
                 fill = "lightsalmon",
                 color = "black")
```



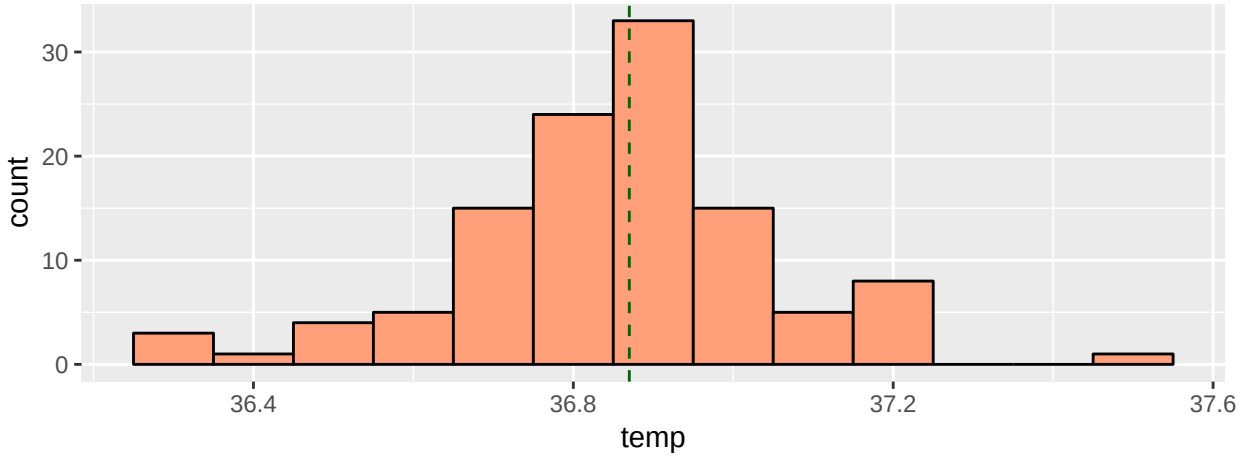
На гистограммы можно добавлять вспомогательные вертикальные или горизонтальные линии. Например, можно отчертить значение частоты, равной 15:

```
# слой geom_hline
# yintercept - значение, где прямая пересекает ось y

ggplot(data = beav, aes(x = temp)) +
  geom_histogram(binwidth = 0.1,
                 fill = "lightsalmon",
                 color = "black") +
  geom_hline(yintercept = 15, color = "darkgreen")
```



Или отметить медиану (уже вертикальная линия):

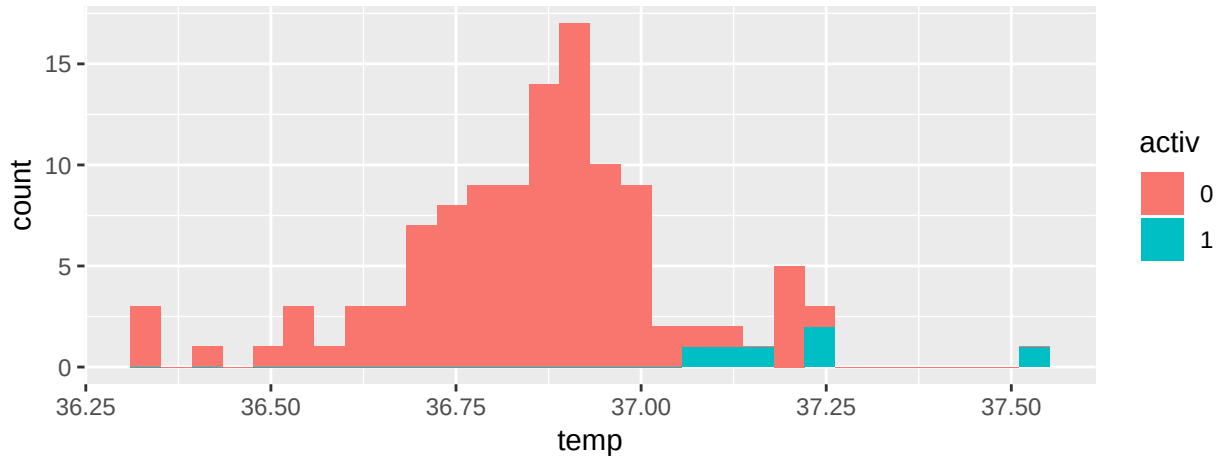
```
# слой geom_vline
# xintercept - значение, где прямая пересекает ось x
# lty - тип линии

ggplot(data = beav, aes(x = temp)) +
  geom_histogram(binwidth = 0.1,
                 fill = "lightsalmon",
                 color = "black") +
  geom_vline(xintercept = median(beav$temp),
             color = "darkgreen",
             lty = 2)
```



Гистограммы тоже можно строить отдельно для разных групп наблюдений, оставаясь при этом в пределах одного графика. В нашем случае это не очень наглядно (в группе активных бобров всего 6 наблюдений), но для примера можно построить гистограммы по группам:

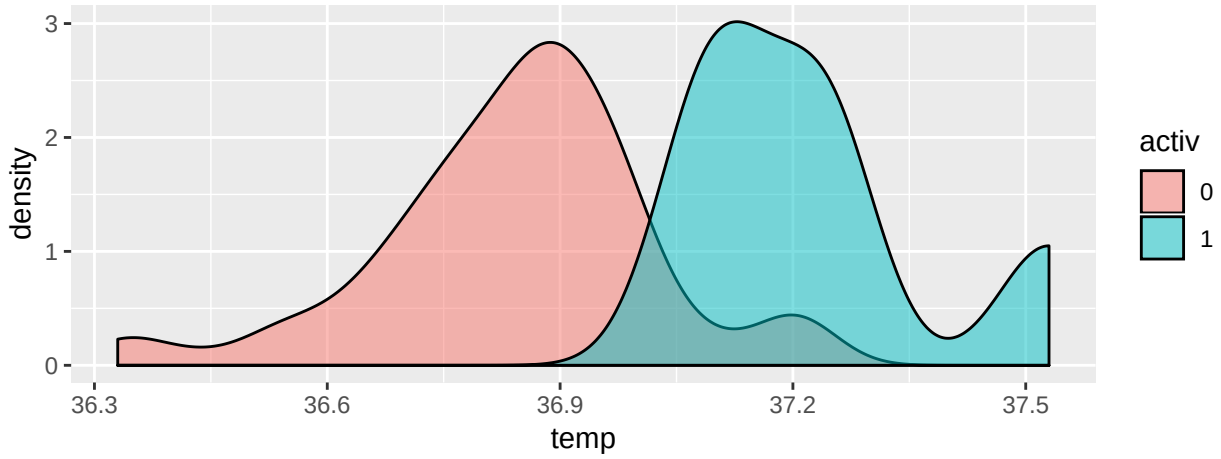
```
ggplot(data = beav, aes(x = temp,
                        group = activ,
                        fill = activ)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Если требуется построить графики распределения, которые накладываются друг на друга, для большей наглядности вместо гистограмм иногда используют сглаженные графики плотности распределения (*kernel density plots*).

```
# alpha = 0.5 - для 50% прозрачности
ggplot(data = beav, aes(x = temp,
                        group = activ,
                        fill = activ)) +
  geom_density(alpha = 0.5)
```

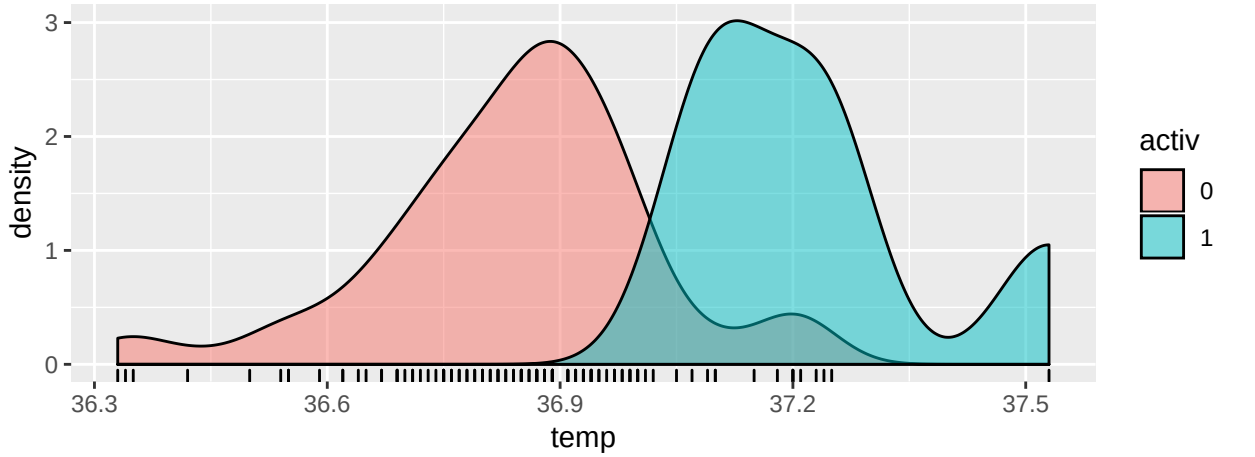


Такие графики выглядят симпатично, однако могут дезинформировать. Из-за того, что такие графики плотности получаются путем сглаживания гистограммы, они могут получаться неточными. Например, не будут отражены некоторые перепады в частотах, сгладятся «пики» распределения. Даже в нашем случае заметны неточности: по гистограмме видно, что в распределении температуры тела активных бобров есть «дырки» (некоторых столбцов нет), а на сглаженном графике плотности мы этого не увидим.

Раз уж зашла речь о дезинформации, обсудим ещё один важный момент. Очень удобно (и честно!), когда на графике по возможности отражено число наблюдений, по которому он строился. Если в случае с точечными графиками это видно и так (много точек или совсем мало), то в случае гистограмм, графиков плотности, ящиков с усами и прочих, число наблюдений определить по графику сложновато. В `ggplot2` для отметки наблюдений есть специальный параметр `rugs` (устоявшегося русскоязычного термина нет). Выглядит это следующим образом:

```
ggplot(data = beav, aes(x = temp,
                        group = activ,
```

```
                      fill = activ)) +
  geom_density(alpha = 0.5) +
  geom_rug()
```

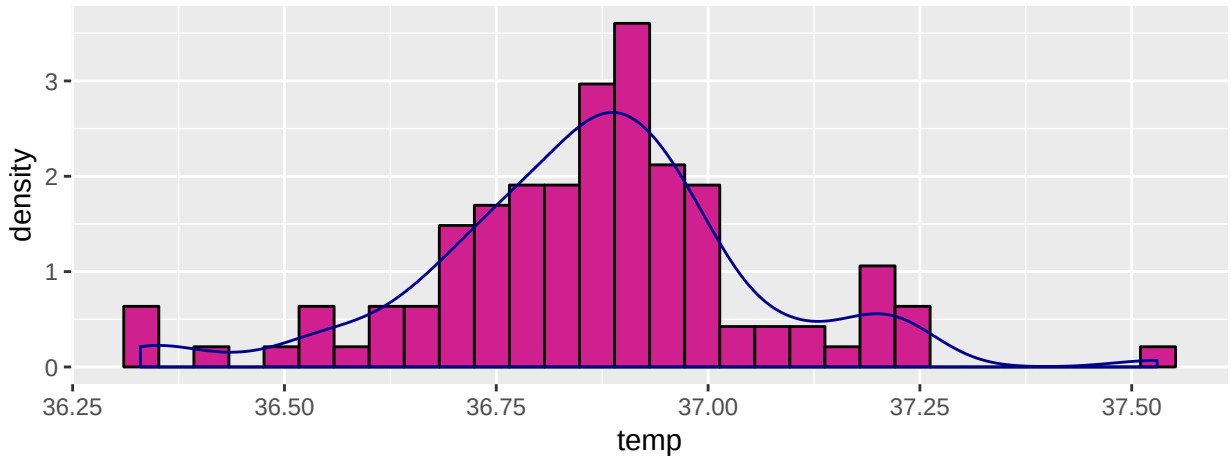


Под графиком добавляются «палочки» — обозначения наблюдений. И хотя эти засечки (*rugs*) не показывают явно общее числе наблюдений (вряд ли кто-то захочет их считать), по ним можно представлять, сколько наблюдений сконцентрировано в той или иной части графика. Зачем это нужно? Представим, что мы ничего не знаем о базе данных по бобрам и видим графики плотностей распределения по группам. Нам может показаться, что, если мы исключим несколько значений температуры тела активных бобров в окрестности 37.5 градусов, распределение температуры тела этих бобров будет похоже на нормальное. Однако, когда мы посмотрим на график с *rugs*, про нормальность мы думать не будем — увидим, что в группе всего 6 наблюдений, а это очень мало, чтобы судить о форме распределения.

Бонус. Как наложить сглаженный график плотности (без заливки) на гистограмму?

```
# y = ..density.. : функция для вычисления плотности,
# обособляется точками

ggplot(data = beav, aes(x = temp)) +
        geom_histogram(aes(y = ..density..),
                       fill = "violetred",
                       color = "black") +
        geom_density(col = "darkblue")
```

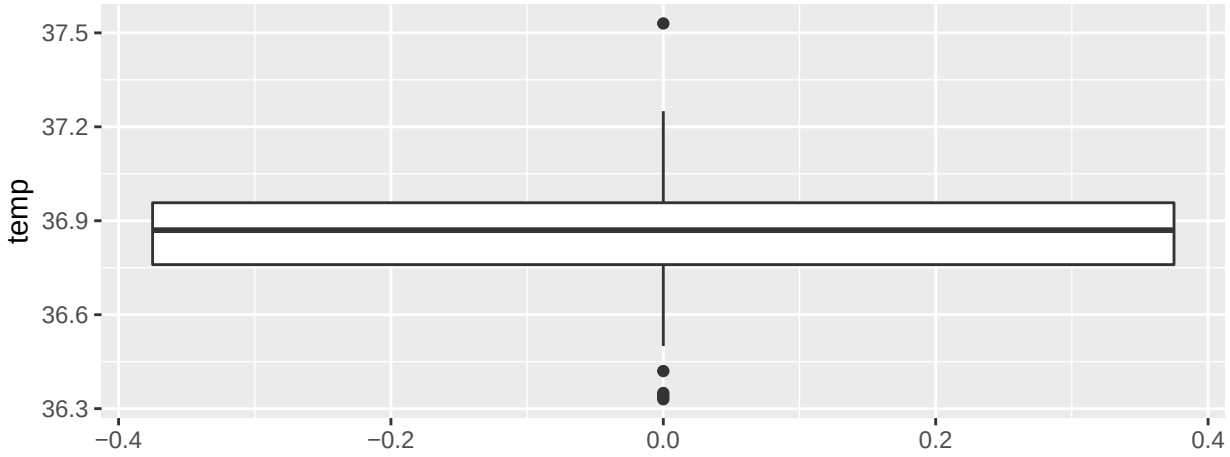
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Ящики с усами (*box plots*) и скрипичные диаграммы (*violin plots*)

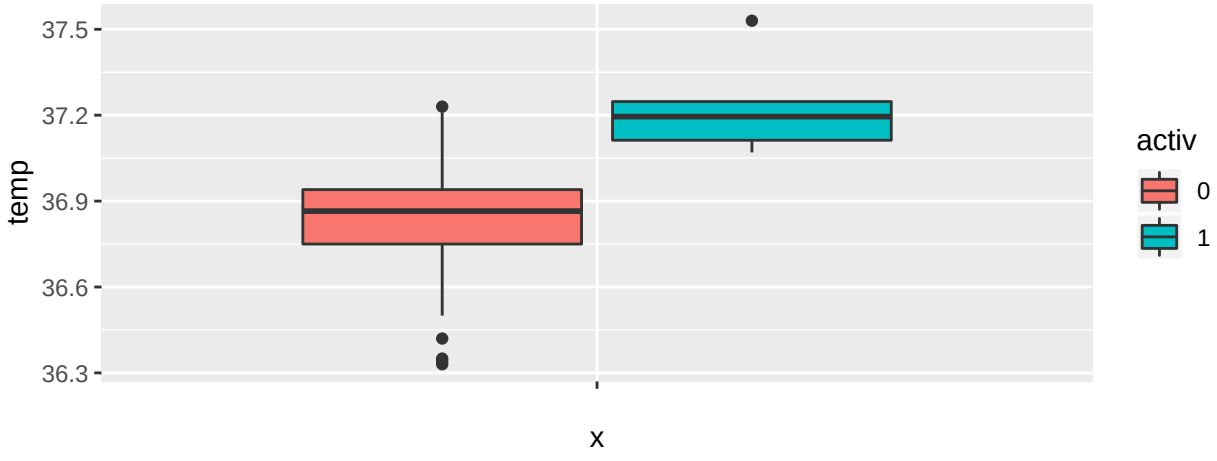
Про ящики с усами и скрипичные диаграммы мы уже говорили, поэтому давайте просто их построим. Просто ящик с усами:

```
# слой geom_boxplot

ggplot(data = beav, aes(y = temp)) +
  geom_boxplot()
```

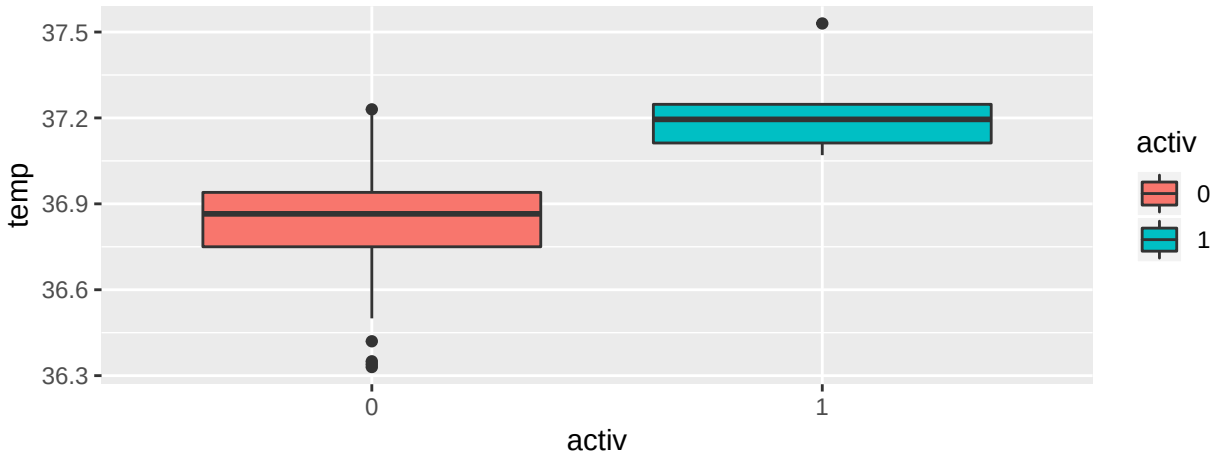


Ящики с усами по группам:

```
# вариант 1, пустая ось x и группировка в group

ggplot(data = beav, aes(x = "", y = temp,
                        group = activ,
                        fill = activ)) +
  geom_boxplot()
```




```
# вариант 2, группировка по оси x

ggplot(data = beav, aes(x = activ, y = temp,
                        fill = activ)) +
  geom_boxplot()
```



Скрипичные диаграммы по группам:

```
# слой geom_violin

ggplot(data = beav, aes(x = "", y = temp,
                        group = activ,
                        fill = activ)) +
  geom_violin()
```