**Алгоритм детекции событий I-VT (velocity threshold)**

На входе имеем последовательность сэмплов, размеченных как “F”, “S” или “G”, где метка F – указатель того, что моментальная скорость меньше или равна заданному порогу VT, метка S – указатель того, что моментальная скорость больше заданного порога VT, метка G – указатель пропуска в данных (GAP: нулевые или выходящие за границы экрана/сцены координаты).

При этом известны параметры VT, angular, screenDist, screenDim, screenSize, maxTimeBetweenFixations, maxDistBetweenFixations, minFixLength, maxGapLength, maxVelocity, maxAcceleration

При этом известно, как оценивать близость двух сэмплов в пространстве и времени: рассчитываем с помощью функции calcPxVel или функции calcAngVel дельту по времени и дельту по расстоянию:

1. if (angular)

{

res <- calcPxVel(t, x, y); dist <- res$dists; dt <- res$dts

}

1. if (!angular)

{

res <- calcAngVel(t, x, y, screenDist, screenDim, screenSize); dist <- res$dists; dt <- res$dts

}

Разбиваем последовательность на группы последовательных сэмплов, соответствующих каждой метке (список groups с элементами – фреймами данных, содержащими время t, координаты x,y, метки группы group, номер группы groupNum). Например, последовательность GGGGSSSFFFFFSSSFFFSSSSSGGGGFFFGSSSSFFFSFGGGSSS будет разбита на 15 групп: 1) GGGG; 2) SSS; 3) FFFFF; 4) SSS; 5) FFF; 6) SSSSS; 7) GGGG; 8) FFF; 9) G; 10) SSSS; 11) FFF; 12) S; 13) F; 14) GGG; 15) SSS.

Создаём пустые переменные-списки fixationGroups, saccadeGroups, gapGroups, artifactGroups. Создаём переменную lastGroup = NA.

Бежим в цикле по номерам групп (x in (1:length(groups)), а внутри цикла:

1. определяем текущую группу currentGroup = groups[[x]]$group[1]
2. если currentGroup = “F”, то:
   1. проверяем, не короткая ли это фиксация, сравнивая длительность группы с порогом minFixLength;
   2. если фиксация короткая, то:
      1. lastGroup не изменяется;
      2. переходим на следующую итерацию к следующей группе;
   3. если фиксация не короткая, то:
      1. формируем группу фиксации fixationGroup = groups[[x]]
      2. если lastGroup = NA, то добавляем fixationGroup в список fixationGroups, lastGroup = “F”;
      3. если lastGroup = “S”, то проверяем, есть ли в списке fixationGroups обнаруженные фиксации; если есть, то проверяем, не близка ли текущая фиксация к предыдущей фиксации по параметрам maxTimeBetweenFixations и maxDistBetweenFixations; если нет, то формируем группу фиксации fixationGroup = groups[[x]] и пополняем список групп фиксаций fixationGroups, lastGroup = “F”;
      4. если близка, то добавляем сэмплы group[[x]] к сэмплам последней группы в списке fixationGroups, опустошаем saccadeGroup, lastGroup = “F”;
      5. если не близка, то формируем группу саккады saccadeGroup = groups[[x-1]] и пополняем список групп саккад saccadeGroups, а также формируем группу fixationGroup = groups[[x]] и пополняем список групп фиксаций fixationGroups, lastGroup = “F”;
      6. если lastGroup = “G”, то формируем группу фиксации fixationGroup = groups[[x]] и пополняем список групп фиксаций fixationGroups, lastGroup = “F”;
      7. переходим на следующую итерацию к следующей группе;
3. если currentGroup = “S”, то:
   1. проверяем по параметрам maxVelocity и maxAcceleration, аномальна ли саккада (включает ли сэмплы с аномальными значениями скорости или ускорения);
   2. если аномальна, то формируем artifactGroup и добавляем artifactGroup в список артефактов artifactGroups, lastGroup не изменяется;
   3. если не аномальна, то формируем группу саккады saccadeGroup = groups[[x]] и пополняем список групп саккад saccadeGroups, lastGroup = “S”;
4. если currentGroup = “G”, то:
   1. проверяем по параметру maxGapLength, длинный ли пропуск или не длинный
   2. если длинный или lastGroup = NA или группа является последней, то формируем группу пропуска gapGroup = groups[[x]] и пополняем список пропусков gapGroups, lastGroup = “G”;
   3. если короткий и группа не является последней, то классифицируем пропуск (по близости сэмплов, смежных с пропуском):
      1. если фиксация, то если lastGroup = “F”, то добавляем сэмплы groups[[x]] к сэмплам последней группы в списке fixationGroups, lastGroup = “F”;
      2. если фиксация, то если lastGroup = “S” или “G”, то формируем группу фиксации fixationGroup = groups[[x]] и добавляем fixationGroup в список fixationGroups, lastGroup = “F”;
      3. если саккада, то если lastGroup = “F” или “G”, то формируем группу саккады saccadeGroup = groups[[x]] и пополняем список групп саккад saccadeGroups, lastGroup = “S”;
      4. если саккада, то если lastGroup = “S”, то добавляем сэмплы groups[[x]] к сэмплам последней группы в списке saccadeGroups, lastGroup = “S”.