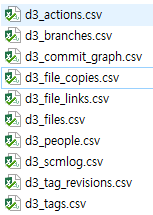
Visual Exploration Exercise Report

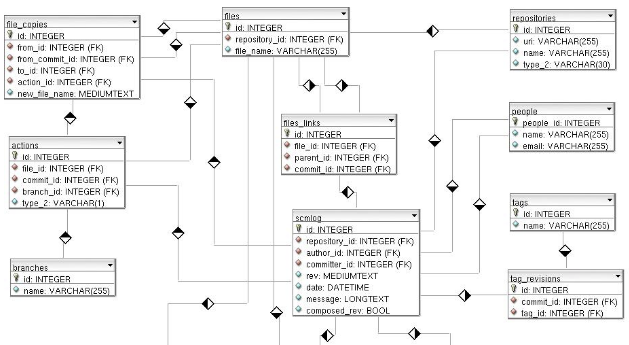
2017-33048 김영택

1. Analytics Tool: Tableau
2. Dataset

* D3의 Git 데이터 (<https://github.com/d3/d3>) 활용
* 추출방법: SCM 분석용 오픈소스인 CVSAnaly (<https://github.com/MetricsGrimoire/CVSAnalY>)를 활용하여 git data를 DB형태로 뽑은 다음 각각의 table을 csv 형태로 변형



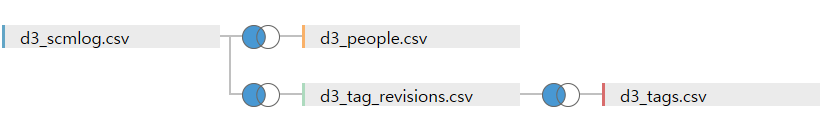
* 데이터셋: 4166개(commit 기반), 10개 sheet(table)



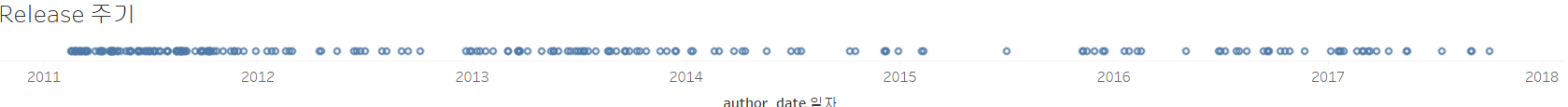
* 주된 Attribute(or Sheet/Table) 설명
  + SCMLOG: 1회의 commit마다 1 row씩 생성
  + People: Author(소스코드의 작성자), Committer
  + Tag: commit에 달려있는 Annotation. 주로 version을 구분할 때 활용된다.
  + Files: commit에 동반된 파일 리스트를 확인할 수 있다.
  + Branch: git의 branch. 기본은 master이다.
  + Actions: commit 별로 file의 추가/수정/삭제 이력이 포함되어 있다.

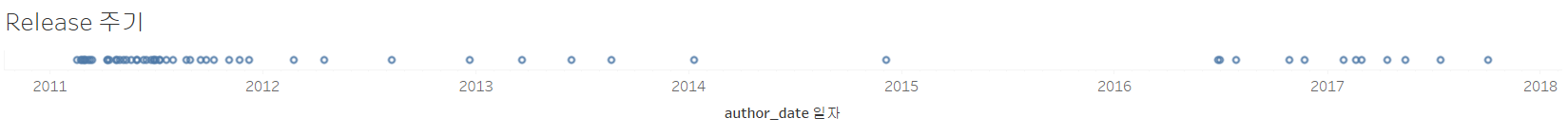
1. Insight/Findings

* 먼저, sheet 구조를 위 database schema에 맞춰서 아래와 같이 설정하였다.

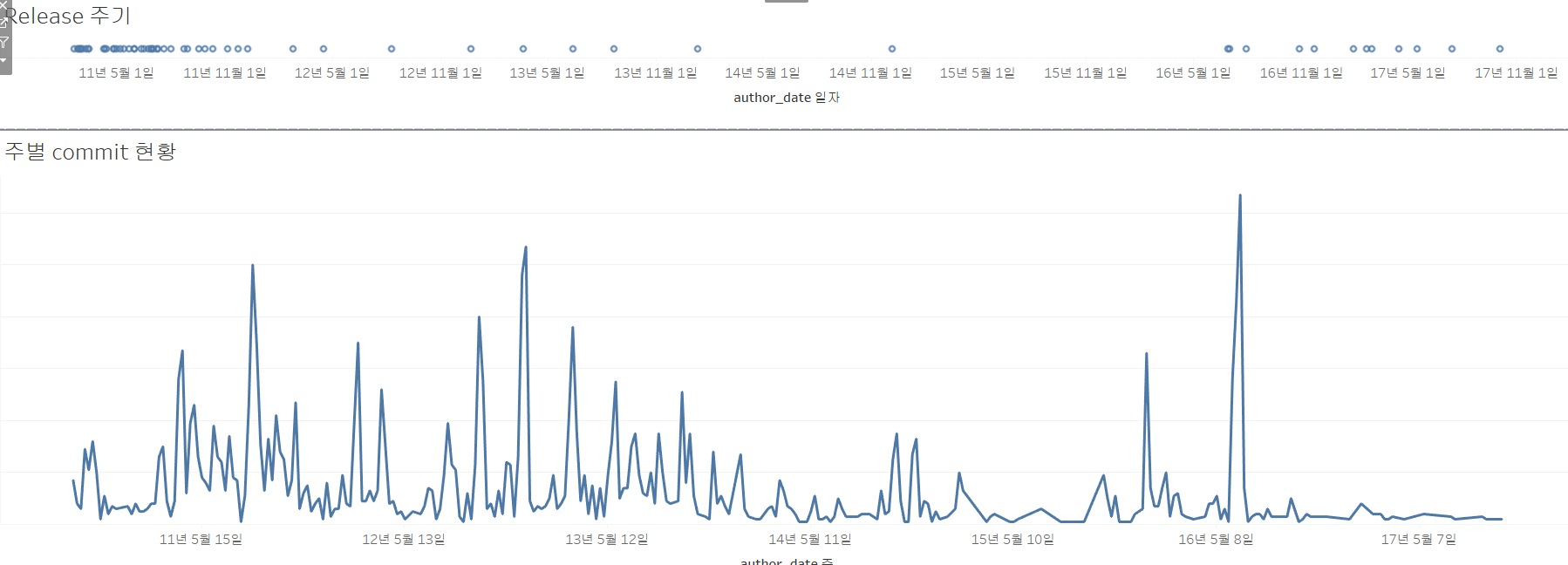


* 1. Release 주기  
     시간에 따른 릴리즈 패턴을 살펴보면, 개발 초기에 가장 많이 집중되어 있고, major 버전이 발행되기 이전에는 뜸한 것을 알 수 있다.   
     그 중 3.0과 4.0 사이의 릴리즈 사이가 많이 없는 걸로 봐서, 3.0이 역대로 안정되게 활용된 버전이란 걸 추측해볼 수 있다.  
     (시간 단위로 bar/line graph로 보는 것 보다 dot으로 표현하여 얼마나 겹치는지를 확인해서 보는게 더 도움이 되었다)



Major 버전 주기  
: 메이저 버전(X.X.0)만 필터링 해서 보면   


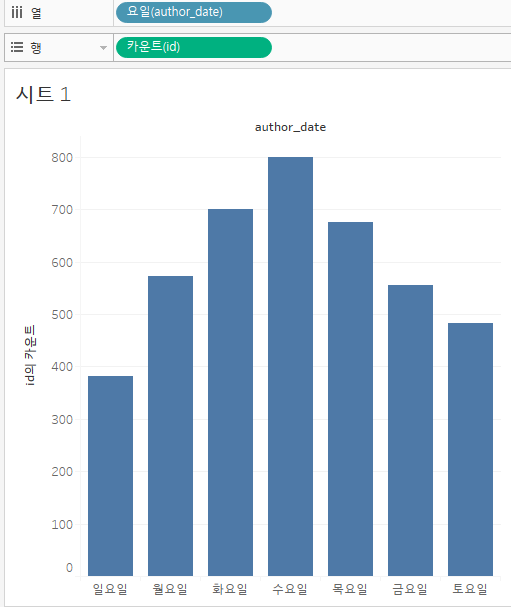
* 1. Major 버전(X.X.0)이 나올 때 개발자 활동이 빈번한지 확인

- tag 정보가 commit 하나마다 묶여있는데,  
NULL을 제외하고 line graph에 하나씩 표현하는 방법을 잘 몰라서  
아래와 같이 release주기 테이블과 commit 현황 그래프를 유사한 scale로 병치시켜서 분석하였다.  


반드시 그렇지만은 않음을 확인하였다.

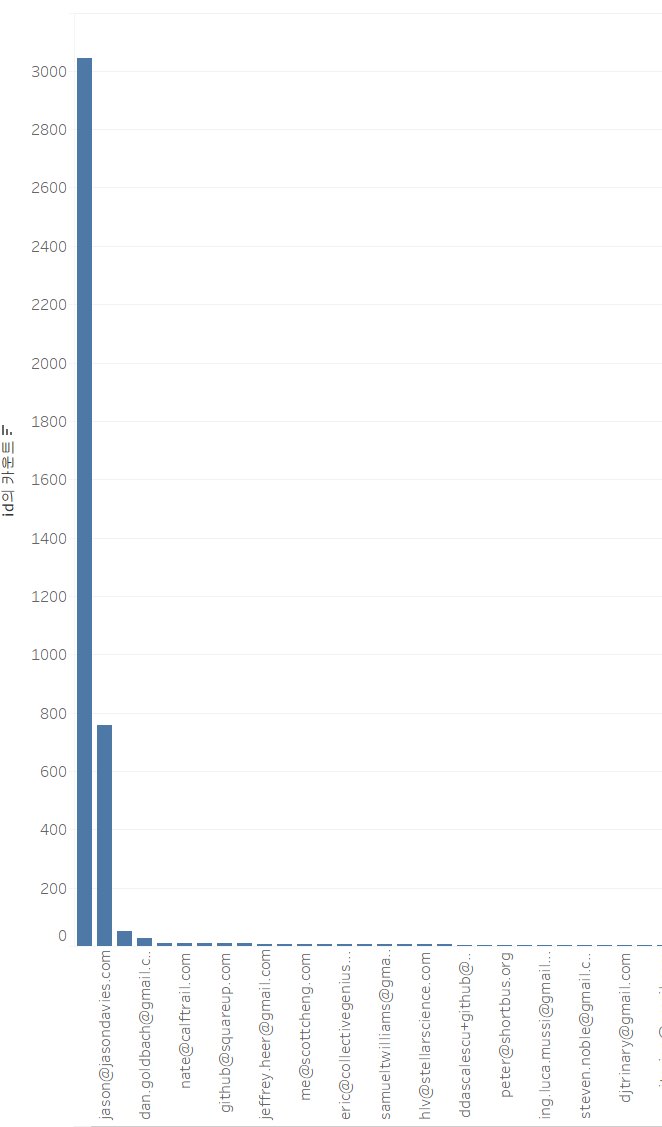
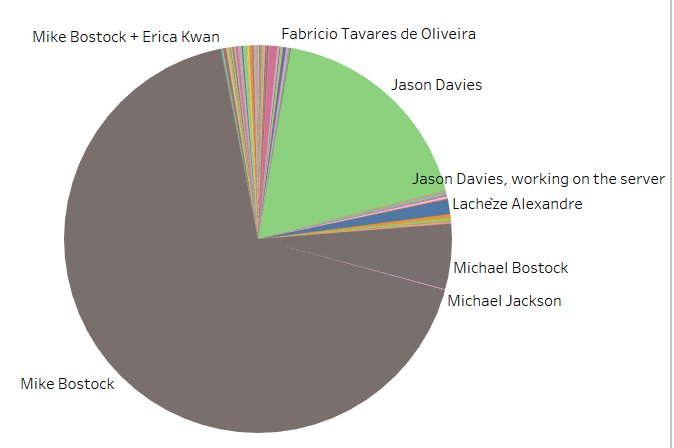
앞자리 버전 변화때는(특히 4.0 릴리즈) 활동이 많으나, 나머지 중간 자리 변화때는 릴리즈 되는 feature에 따라 양이 적기도, 많기도 한 듯 하다.

* 1. D3 개발자들도 주말보다는 주중에 작업을 더 많이 한다!  
     하지만, 주말에도 쉬지는 않는다 🡺 야근/특근 많은 회사원 타입

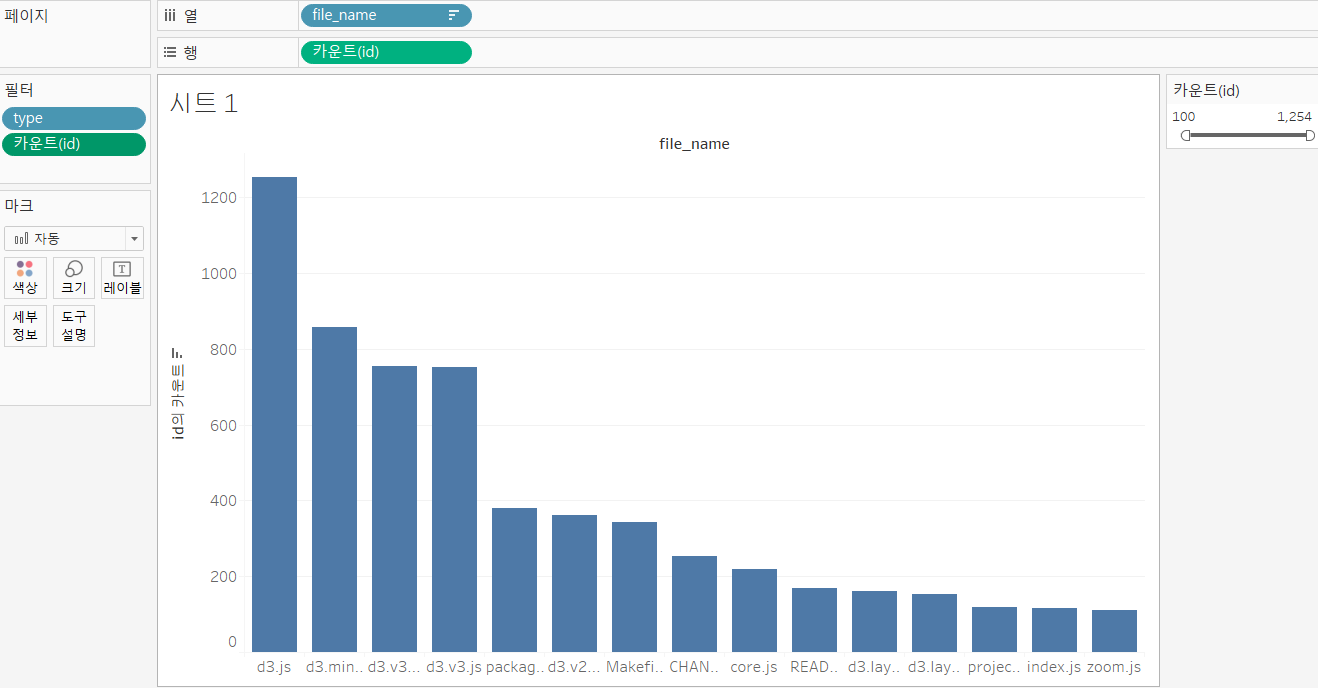


* 1. 주 개발자는 Mike Bostock과 Jason Davies 이다. 2명이 완전히 dominant하다.

ordering해보면 완전히 right skew 되어 있음을 알 수 있다.

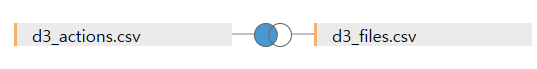


* 1. 가장 많이 수정된 파일일수록 버그가 많을 가능성이 많거나, change impact이 클 가능성이 높으므로, 아래와 같은 차트 조건을 그려 살펴보았다.  
     : action type을 M(modified)로 필터를 걸고, action 테이블과 file테이블을 조인하여 M이 많이 발생하는 file name별로 정렬함



당연히 외부에서 불러서 사용하는 d3.\*.js 파일이 높았다.

결과를 위해서 테이블은 아래와 같이 조인하였다.



1. Discussion

* 테이블 간의 join이 생각보다 GUI로 쉽게 조작할 수 있어서, 다수 테이블의 관계를 잘 볼 수 있었다.
* 전체적으로 기존의 SQL을 사용해서 데이터를 뽑은 후 excel로 차트를 만드는 것보다 훨씬 빠르고 쉽게 차트를 그릴 수 있었다.
* 테이블을 병치해서 보기 위해서 대쉬보드를 반드시 만들어야 하는 것이 조금 불편했다. – spotfire로 하는게 이런건 더 유용할 듯 하다.
* Tree (node-diagram) 구조를 그려보고 싶었으나, 많이 복잡하여서 현재 테크닉 수준으로는 그리기가 좀 어려웠다. (<https://community.tableau.com/thread/154623>)   
  tree로 commit 과 branch간의 관계를 살펴보았다면 더 다양한 분석을 할 수 있었을 듯 하다.