Exploratory Data Analysis

Christian Birchler

5/21/2020

Load data and packages

```
library(data.table)
library(ggplot2)
library(grid)
library(gridExtra)
library(ggbiplot)
```

```
## Loading required package: plyr
```

```
## Loading required package: scales
```

```
set.seed(123)

########## READ DATA ##########
setwd("~/Desktop/bsc-analysis")
dd.orig <- fread("vm6-no-gc-merged.csv")
idflakies <- fread("idflakies-projects.csv")
################################
```

Clean data

A data set with roughly 400'000 entries by 74 variable has probably some invalid (or corrupted) data entries. First of all they must be identrified and treated separately.

```
dd <- dd.orig

# convert all collumns with numbers as numeric data type
# collect indices of NA entries -> those are corrupted data entries
dd$deadlock.count <- as.numeric( dd$deadlock.count)
```

```
## Warning: NAs introduced by coercion
```

```
na.indeces <- which( is.na( dd$deadlock.count))

dd$loaded <- as.numeric( dd$loaded)
```

```
## Warning: NAs introduced by coercion
```

```
na.indeces <- unique( c(na.indeces, which( is.na(dd$loaded))))

dd$pools.Metaspace.used <- as.numeric( dd$pools.Metaspace.used)
```

```
## Warning: NAs introduced by coercion
```

```
na.indeces <- unique( c(na.indeces, which( is.na(dd$pools.Metaspace.used))))

dd$`pools.PS-Eden-Space.committed` <- as.numeric(dd$`pools.PS-Eden-Space.committed`)
```

```
## Warning: NAs introduced by coercion
```

```
na.indeces <- unique( c(na.indeces, which( is.na(dd$`pools.PS-Eden-Space.committed`))))

dd$`pools.PS-Eden-Space.init` <- as.numeric(dd$`pools.PS-Eden-Space.init`)
```

```
## Warning: NAs introduced by coercion
```

```
na.indeces <- unique( c(na.indeces, which( is.na(dd$`pools.PS-Eden-Space.init`))))

dd$`pools.PS-Eden-Space.max` <- as.numeric(dd$`pools.PS-Eden-Space.max`)
```

```
## Warning: NAs introduced by coercion
```

```
na.indeces <- unique( c(na.indeces, which( is.na(dd$`pools.PS-Eden-Space.max`))))

dd$`pools.PS-Eden-Space.used` <- as.numeric(dd$`pools.PS-Eden-Space.used`)
```

```
## Warning: NAs introduced by coercion
```

```
na.indeces <- unique( c(na.indeces, which( is.na(dd$`pools.PS-Eden-Space.used`))))

dd$`pools.PS-Survivor-Space.committed` <- as.numeric(dd$`pools.PS-Survivor-Space.committed`)
```

```
## Warning: NAs introduced by coercion
```

```
na.indeces <- unique( c(na.indeces, which( is.na(dd$`pools.PS-Survivor-Space.committed`))))

dd$terminated.count <- as.numeric(dd$terminated.count)
```

```
## Warning: NAs introduced by coercion
```

```
na.indeces <- unique( c(na.indeces, which( is.na(dd$terminated.count))))

# proportion of corrupted data entries -> ~1%
print(length( na.indeces)/nrow(dd))
```

```
## [1] 0.009638677
```

```
# omit corrupted data entries
dd <- dd[-na.indeces,]
```

Flakiness

```
unique_tests <- unique(dd[, c('TestCase','ProjectName','CommitHash')])

# count the different outcomes per test case (group by TestCase,ProjectName and CommitHash)
nr_different_outcomes <- dd[  , list(`pass/fail` = `pass/fail`, Count = uniqueN(`pass/fail`))
                              , by=c('TestCase','ProjectName','CommitHash') ]
different_outcomes <- nr_different_outcomes[ Count > 1 ,  ]
```

```
# less than 20 entries -> build failure -> flaky?
nr_measurements_per_test <- dd[  , list(Count = .N)
                                 , by=c('TestCase','ProjectName','CommitHash') ]

# there are only even number of test measurements of a single test case
# --> there can only build failures if there are less than 20 measurements
less_than_20_entries <- nr_measurements_per_test[ Count < 20 ,  ] # flaky?

# merge (union) different outcomes with tests which have not 20 measurements
different_outcomes_or_less_20_entries <- merge(different_outcomes, less_than_20_entries
                                               , by=c('TestCase','ProjectName','CommitHash')
                                               , all=TRUE)

#flaky.own <- different_outcomes_or_less_20_entries
flaky.own <- unique( different_outcomes[, c('TestCase','ProjectName','CommitHash')] )

# percentage of flaky tests in own measurements
flakiness_rate <- nrow(flaky.own)/nrow(unique_tests)

flakiness_rate
```

```
## [1] 0.01381114
```

Here is to consider the definition of flakiness (`flaky.own`). In the current setting I consider a test as flaky if there are different outcomes (pass/failure/error) of the test. I do not consider the case where the test code compilation failed for some but not all iterations.

From all the measurements were 1.3% of the tests flaky according to the definition above.

Comparison with idFlakies data set

```
# set merge with idflakies data set (intersection over 'TestCase','CommitHash')
flaky.intersect <- merge(flaky.own, idflakies, by.x=c('TestCase','CommitHash')
                         , by.y=c('Test Name','SHA'), all=FALSE)

nrow(flaky.intersect)
```

```
## [1] 2
```

`flaky.intersect` has only two entries. This means that the recorded flaky tests of the idFlakies data set could not be well replicated. Only two of their flaky tests could be replicated. Nevertheless, from the own measurements multiple additional flaky tests could be identified.

Summary of all software projects

Below is a list of all projects where measurements were taken and the number of flaky and non-flaky tests are also listed.

```
# label data set (flaky/not flaky)
flaky.own$isFlaky <- TRUE
dd.labeled <- merge(dd, flaky.own, by=c('TestCase','ProjectName','CommitHash')
                    , all=TRUE)
```

```r
dd.labeled[ is.na(isFlaky) , "isFlaky"] <- FALSE

# unique tests cases (original data set has before/after entries)
labeled_tests <- unique( dd.labeled[, c('TestCase','ProjectName','CommitHash','isFlaky')] )



flaky_tests_per_project <- labeled_tests[ isFlaky == TRUE , list(NrFlaky = .N)
                                          , by=c('ProjectName','CommitHash') ]
non_flaky_tests_per_project <- labeled_tests[ isFlaky == FALSE , list(NrNotFlaky = .N)
                                              , by=c('ProjectName','CommitHash') ]

projects_overview <- merge(flaky_tests_per_project, non_flaky_tests_per_project,
                           by=c('ProjectName','CommitHash'), all=TRUE)

projects_overview[,-'CommitHash']
```

```
##                     ProjectName NrFlaky NrNotFlaky
##  1:                    Activiti       2        171
##  2:                  GoodData-CL      NA         21
##  3:              Java-WebSocket       1        144
##  4:                      Struts      64        268
##  5:                WebCollector      NA          9
##  6:                     admiral      NA        136
##  7:                  aismessages      NA         44
##  8:                     aletheia      NA         31
##  9:                      as2-lib      NA         47
## 10:                        c2mon       3        372
## 11:                        cukes      NA         54
## 12:                db-scheduler      NA         44
## 13:     delight-nashorn-sandbox      NA         77
## 14:            elastic-job-lite      NA        553
## 15:                       esper      NA          4
## 16:                  excelastic      NA         11
## 17:          fluent-logger-java      NA         16
## 18:                      hadoop      NA        671
## 19:                      helios       1        205
## 20:      hsac-fitnesse-fixtures      NA        250
## 21:                      hutool      NA        508
## 22:             incubator-dubbo      10       1509
## 23:             java-cas-client      NA        150
## 24:                   jfreechart      NA       2176
## 25:           jhipster-registry       4         49
## 26:                       luwak      NA        160
## 27:                  marine-api      NA        925
## 28:       nexus-repository-helm      NA         18
## 29:               nifi-registry      NA        251
## 30:                       noxy      NA         29
## 31:                oci-java-sdk      NA        112
## 32:                        one      NA         12
## 33:                      orbit       1        184
## 34:                       oryx      20        372
## 35:                      pippo      NA        207
## 36:                   querydsl      16       1841
```

```
## 37:                     redpipe      NA         61
## 38:                    retrofit      NA        605
## 39:                     riptide       8        315
## 40:               rxjava2-extras      NA        385
## 41:                     sawmill      NA          2
## 42:                         sos      NA          4
## 43:             spring-cloud-aws      NA        296
## 44: spring-cloud-zuul-ratelimit      NA          3
## 45:            spring-data-ebean      NA         48
## 46:           spring-data-envers      NA         10
## 47:                    spring-ws      NA        109
## 48:                       timely      72         52
## 49:                        tyrus      38        491
## 50:                       unix4j      NA        428
## 51:    vertx-completable-future      NA         61
## 52:       vertx-rabbitmq-client       1          5
## 53:                        whois      NA       2540
## 54:             wikidata-toolkit      NA        786
## 55:                      wildfly      16        118
## 56:        wildfly-maven-plugin      NA         18
## 57:                        wro4j       7        861
## 58:                         yawp      NA         22
##                     ProjectName NrFlaky NrNotFlaky
```

Projects of interest

According to the table above it might be more interesting to focus only on projects with a fair amount of flaky tests to gain a better understanding. It does not make really sense to investigate measurements with no flaky tests or only a few. Below you see the depicted projects of interest.

```
projects_overview[c(4,25,34,39,55), -'CommitHash']
```

```
##          ProjectName NrFlaky NrNotFlaky
## 1:            Struts      64        268
## 2: jhipster-registry       4         49
## 3:              oryx      20        372
## 4:           riptide       8        315
## 5:           wildfly      16        118
```

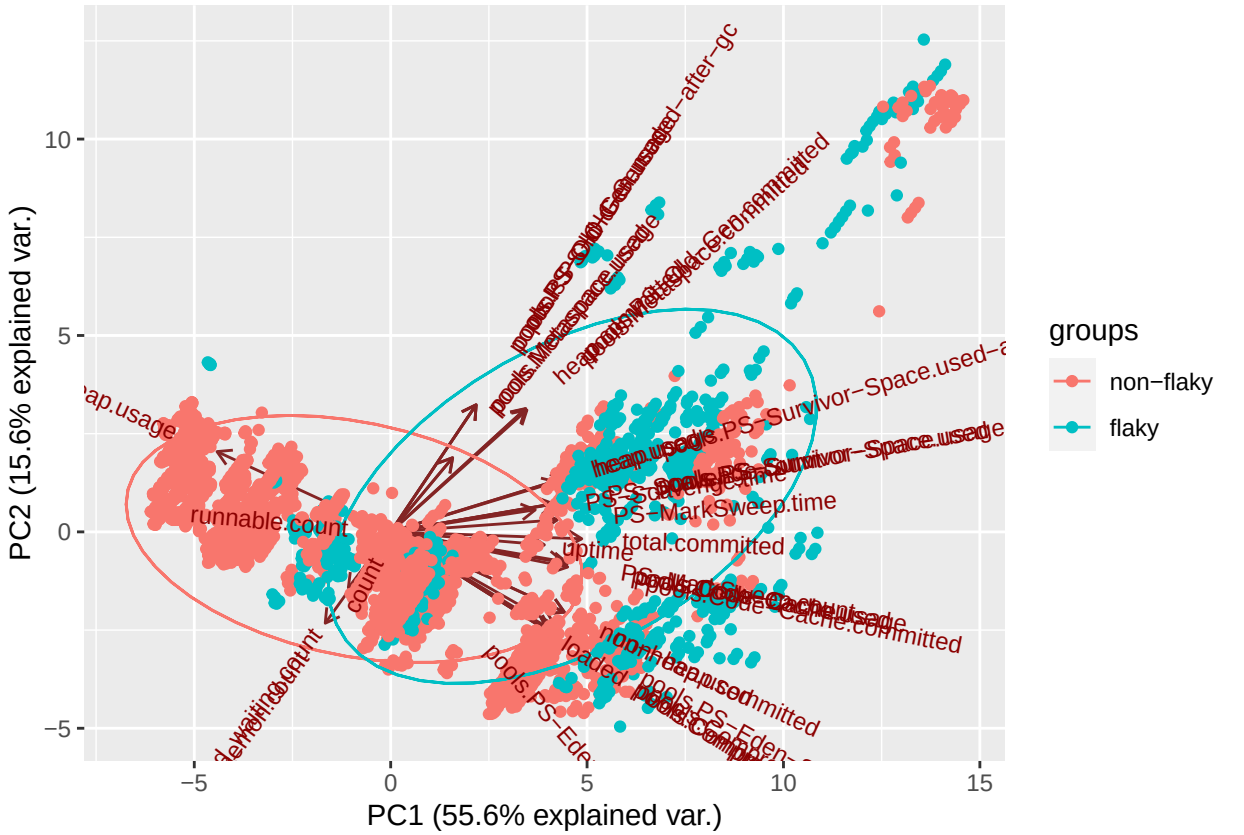
PCA dimensionality reduction

Since the data set is multivariate it makes sence to do a PCA to see if we can position flaky tests in a region where the data has a high variance. For this purpose I treat each project (listed above) separately.

Struts

```
struts <- dd.labeled[ProjectName == 'Struts',]
# struts$deadlock.count <-as.numeric(struts$deadlock.count)
# struts$loaded <- as.numeric(struts$loaded)
# struts$pools.Metaspace.used <- as.numeric(struts$pools.Metaspace.used)
```

```r
# struts$`pools.PS-Eden-Space.committed` <- as.numeric(struts$`pools.PS-Eden-Space.committed`)
# struts$`pools.PS-Eden-Space.init` <- as.numeric(struts$`pools.PS-Eden-Space.init`)
# struts$`pools.PS-Eden-Space.max` <- as.numeric(struts$`pools.PS-Eden-Space.max`)
# struts$`pools.PS-Eden-Space.used` <- as.numeric(struts$`pools.PS-Eden-Space.used`)
# struts$`pools.PS-Survivor-Space.committed` <- as.numeric(struts$`pools.PS-Survivor-Space.committed`)
# struts$terminated.count <- as.numeric(struts$terminated.count)

# waiting count data might be corrupted
#struts$waiting.count <- as.numeric(struts$waiting.count)


#struts[,-c('V1','pass/fail','iteration','TestCase','CommitHash','before/after','ProjectName','deadlock

ignore <- c('V1','pass/fail','iteration','TestCase','CommitHash','before/after','ProjectName','deadlock
select_columns <- c()

# for-loop cannot handle data.table :-(
struts <- as.data.frame(struts)
# PCA cannot be performed if there are vectors if uniform entries
for (i in colnames(struts)){
  x <- struts[3,i]
  if(all(struts[,i] == x) || i %in% ignore){
    next
  }
  select_columns <- c(select_columns, i)
}

#struts <- as.data.table(struts)
#struts[,select_columns]

# perform PCA with scaling and center (avoid missleading eigenvectors)
struts.pca <- prcomp(struts[,select_columns], center=TRUE, scale.=TRUE )

groups <- as.factor(struts$isFlaky)
levels(groups) <- c("non-flaky","flaky")
ggbiplot(struts.pca, groups=groups, ellipse=TRUE, obs.scale=TRUE, var.scale=TRUE)
```

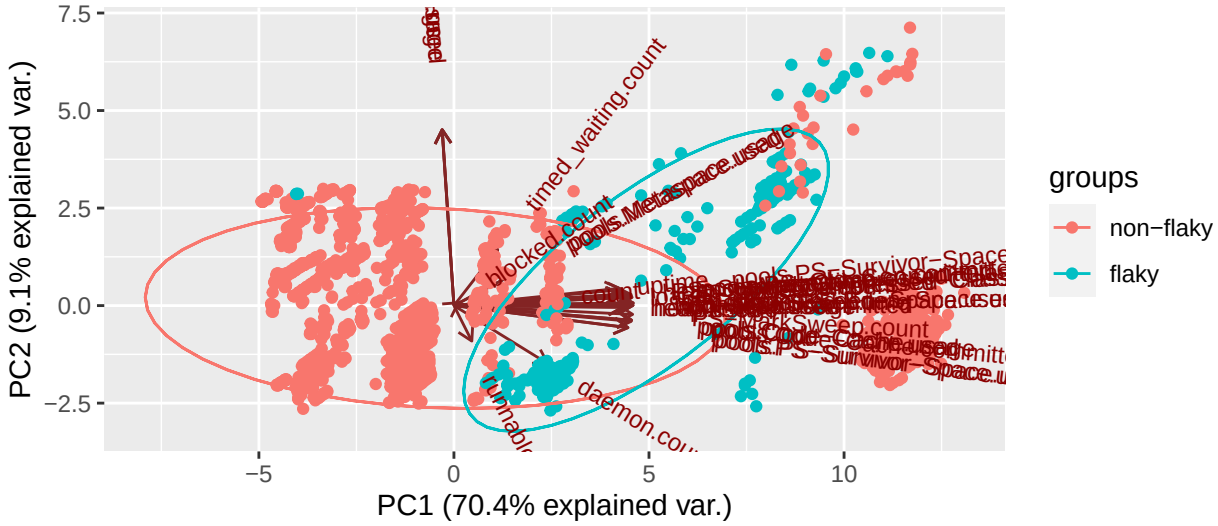


This image shows that the identified flaky tests of 'Struts' can be located separately (more or less) within this biplot.

Wildfly

```
wildfly <- dd.labeled[ProjectName == 'wildfly',]
# wildfly$deadlock.count <-as.numeric(wildfly$deadlock.count)
# wildfly$loaded <- as.numeric(wildfly$loaded)
# wildfly$pools.Metaspace.used <- as.numeric(wildfly$pools.Metaspace.used)
# wildfly$`pools.PS-Eden-Space.committed` <- as.numeric(wildfly$`pools.PS-Eden-Space.committed`)
# wildfly$`pools.PS-Eden-Space.init` <- as.numeric(wildfly$`pools.PS-Eden-Space.init`)
# wildfly$`pools.PS-Eden-Space.max` <- as.numeric(wildfly$`pools.PS-Eden-Space.max`)
# wildfly$`pools.PS-Eden-Space.used` <- as.numeric(wildfly$`pools.PS-Eden-Space.used`)
# wildfly$`pools.PS-Survivor-Space.committed` <- as.numeric(wildfly$`pools.PS-Survivor-Space.committed`)
# wildfly$terminated.count <- as.numeric(wildfly$terminated.count)

# waiting count data might be corrupted
#struts$waiting.count <- as.numeric(struts$waiting.count)


#struts[,-c('V1','pass/fail','iteration','TestCase','CommitHash','before/after','ProjectName','deadlock

#ignore <- c('V1','pass/fail','iteration','TestCase','CommitHash','before/after','ProjectName','deadloc
select_columns <- c()

# for-loop cannot handle data.table :-(
```

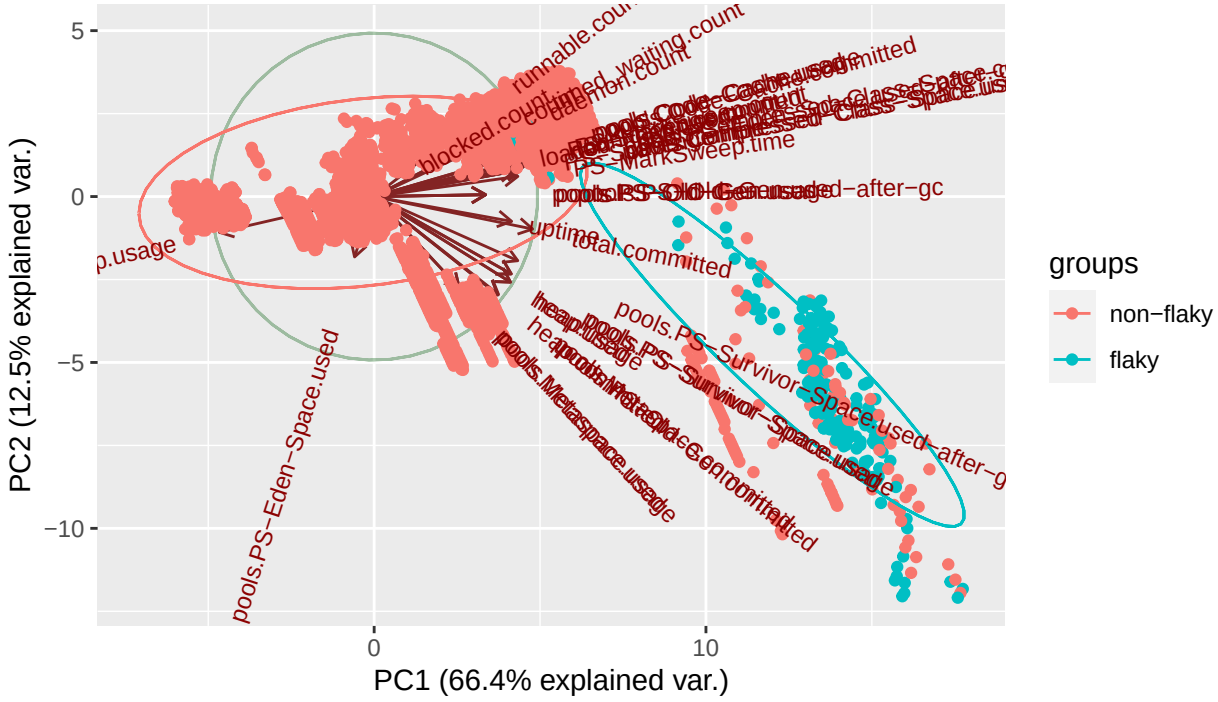
```
wildfly <- as.data.frame(wildfly)
# PCA cannot be performed if there are vectors if uniform entries
for (i in colnames(wildfly)){
  x <- wildfly[3,i]
  if(all(wildfly[,i] == x) || i %in% ignore){
    next
  }
  select_columns <- c(select_columns, i)
}

#struts <- as.data.table(struts)
#struts[,select_columns]

# perform PCA with scaling and center (avoid missleading eigenvectors)
wildfly.pca <- prcomp(wildfly[,select_columns], center=TRUE, scale.=TRUE )

groups <- as.factor(wildfly$isFlaky)
levels(groups) <- c("non-flaky","flaky")
ggbiplot(wildfly.pca, groups=groups, ellipse=TRUE, obs.scale=TRUE, var.scale=TRUE)
```



Oryx

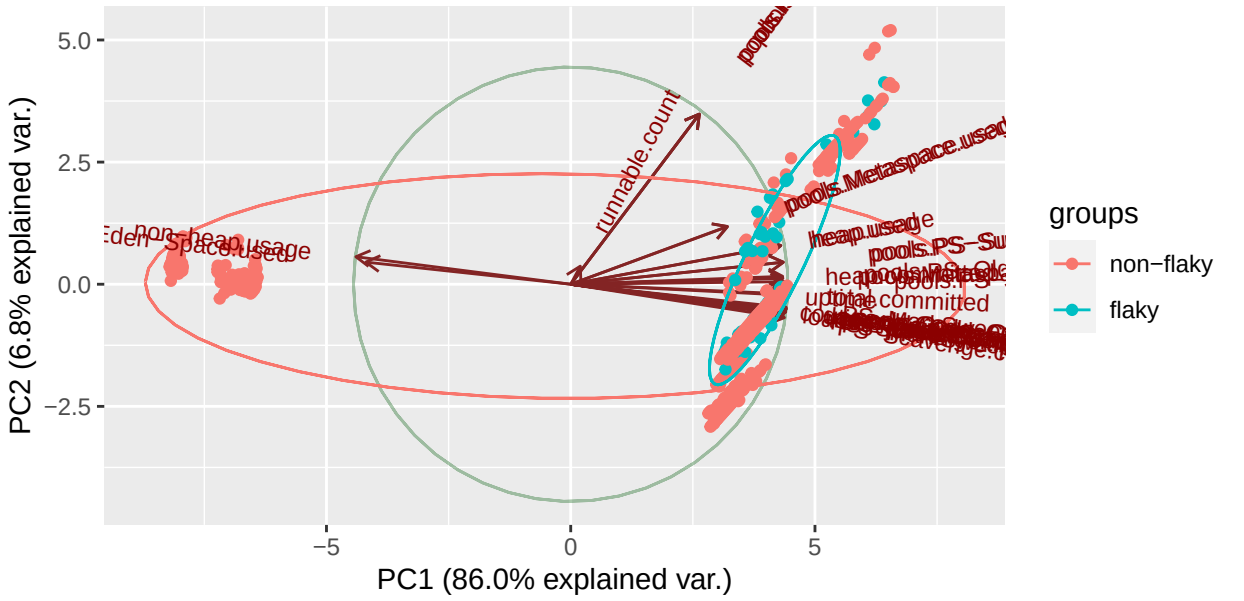
```
oryx <- dd.labeled[ProjectName == 'oryx',]
# oryx$deadlock.count <-as.numeric(oryx$deadlock.count)
# oryx$loaded <- as.numeric(oryx$loaded)
# oryx$pools.Metaspace.used <- as.numeric(oryx$pools.Metaspace.used)
# oryx$`pools.PS-Eden-Space.committed` <- as.numeric(oryx$`pools.PS-Eden-Space.committed`)
# oryx$`pools.PS-Eden-Space.init` <- as.numeric(oryx$`pools.PS-Eden-Space.init`)
# oryx$`pools.PS-Eden-Space.max` <- as.numeric(oryx$`pools.PS-Eden-Space.max`)
# oryx$`pools.PS-Eden-Space.used` <- as.numeric(oryx$`pools.PS-Eden-Space.used`)
# oryx$`pools.PS-Survivor-Space.committed` <- as.numeric(oryx$`pools.PS-Survivor-Space.committed`)
# oryx$terminated.count <- as.numeric(oryx$terminated.count)

# waiting count data might be corrupted
#struts$waiting.count <- as.numeric(struts$waiting.count)
```

```r
#struts[,-c('V1','pass/fail','iteration','TestCase','CommitHash','before/after','ProjectName','deadlock

#ignore <- c('V1','pass/fail','iteration','TestCase','CommitHash','before/after','ProjectName','deadloc
select_columns <- c()

# for-loop cannot handle data.table :-(
oryx <- as.data.frame(oryx)
# PCA cannot be performed if there are vectors if uniform entries
for (i in colnames(oryx)){
  x <- oryx[3,i]
  if(all(oryx[,i] == x) || i %in% ignore){
    next
  }
  select_columns <- c(select_columns, i)
}

#struts <- as.data.table(struts)
#struts[,select_columns]

# perform PCA with scaling and center (avoid missleading eigenvectors)
oryx.pca <- prcomp(oryx[,select_columns], center=TRUE, scale.=TRUE )

groups <- as.factor(oryx$isFlaky)
levels(groups) <- c("non-flaky","flaky")
ggbiplot(oryx.pca, groups=groups, ellipse=TRUE, obs.scale=TRUE, var.scale=TRUE, circle=TRUE)
```



jhipster-registry

```
jhipsterregistry <- dd.labeled[ProjectName == 'jhipster-registry',]
# jhipsterregistry$deadlock.count <-as.numeric(jhipsterregistry$deadlock.count)
# jhipsterregistry$loaded <- as.numeric(jhipsterregistry$loaded)
# jhipsterregistry$pools.Metaspace.used <- as.numeric(jhipsterregistry$pools.Metaspace.used)
# jhipsterregistry$`pools.PS-Eden-Space.committed` <- as.numeric(jhipsterregistry$`pools.PS-Eden-Space.
# jhipsterregistry$`pools.PS-Eden-Space.init` <- as.numeric(jhipsterregistry$`pools.PS-Eden-Space.init`
# jhipsterregistry$`pools.PS-Eden-Space.max` <- as.numeric(jhipsterregistry$`pools.PS-Eden-Space.max`)
# jhipsterregistry$`pools.PS-Eden-Space.used` <- as.numeric(jhipsterregistry$`pools.PS-Eden-Space.used`
# jhipsterregistry$`pools.PS-Survivor-Space.committed` <- as.numeric(jhipsterregistry$`pools.PS-Survivo
# jhipsterregistry$terminated.count <- as.numeric(jhipsterregistry$terminated.count)

# waiting count data might be corrupted
#struts$waiting.count <- as.numeric(struts$waiting.count)


#struts[,-c('V1','pass/fail','iteration','TestCase','CommitHash','before/after','ProjectName','deadlock

#ignore <- c('V1','pass/fail','iteration','TestCase','CommitHash','before/after','ProjectName','deadloc
select_columns <- c()

# for-loop cannot handle data.table :-(
jhipsterregistry <- as.data.frame(jhipsterregistry)
# PCA cannot be performed if there are vectors if uniform entries
for (i in colnames(jhipsterregistry)){
  x <- jhipsterregistry[3,i]
  if(all(jhipsterregistry[,i] == x) || i %in% ignore){
    next
  }
  select_columns <- c(select_columns, i)
}

#struts <- as.data.table(struts)
#struts[,select_columns]

# perform PCA with scaling and center (avoid missleading eigenvectors)
jhipsterregistry.pca <- prcomp(jhipsterregistry[,select_columns], center=TRUE, scale.=TRUE )

groups <- as.factor(jhipsterregistry$isFlaky)
levels(groups) <- c("non-flaky","flaky")
ggbiplot(jhipsterregistry.pca, groups=groups, ellipse=TRUE, obs.scale=TRUE, var.scale=TRUE, circle=TRUE
```

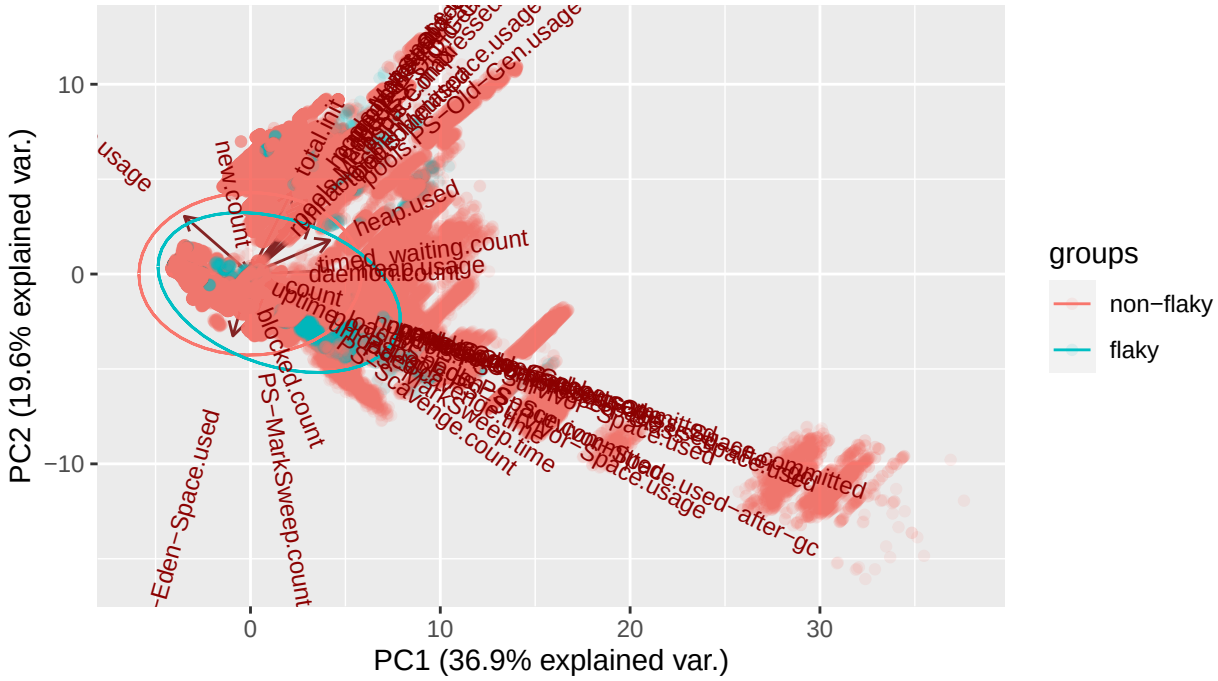


This biplot does not show a clear separat cluster of flkay tests. The flaky tests are evenly distribute among non flaky tests on the right side. It might be that for this project too few data are available in comparison to the previous three projects and their PCA biplot.

PCA of all all measurements together

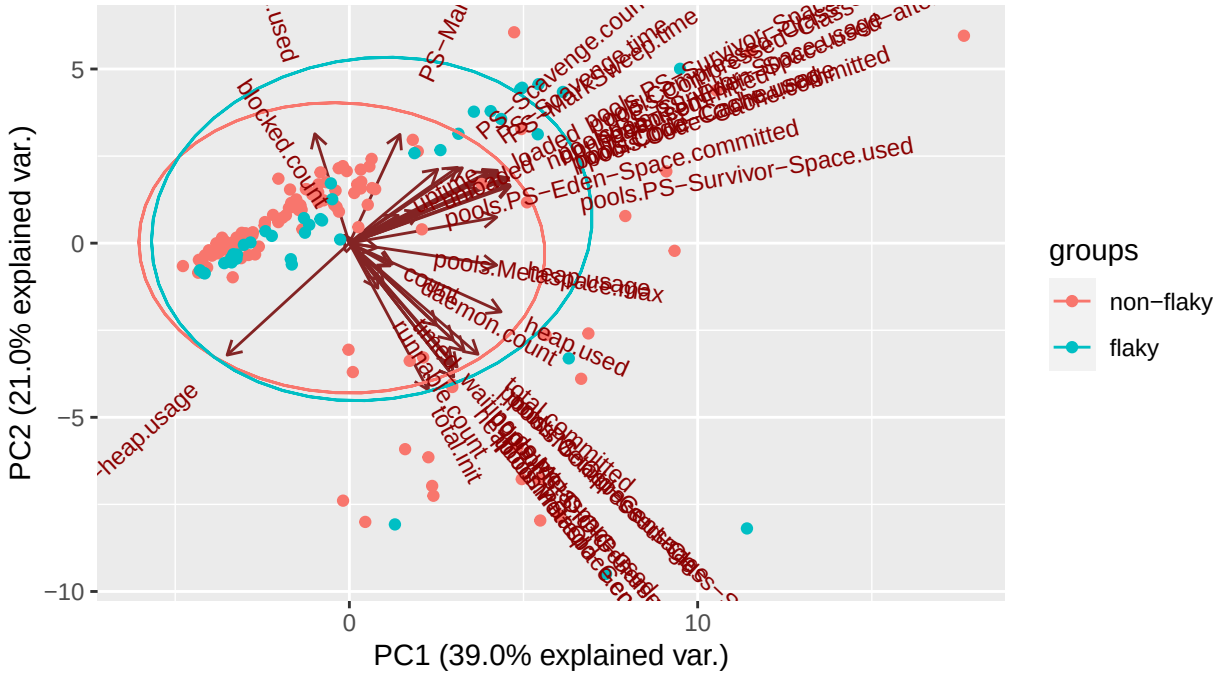
Now we take a look how the biplots look like if we do not separate the projects. For this purpose it makes sence also to scale (as usual) the data for the PCA.

```
# convert all collumns with numbers as numeric data type
# collect indices of NA entries -> those are corrupted data entries
all_together <- dd.labeled

select_columns <- c()

# for-loop cannot handle data.table :-(
all_together <- as.data.frame(all_together)
# PCA cannot be performed if there are vectors if uniform entries
for (i in colnames(all_together)){
  x <- all_together[3,i]
  if(all(all_together[,i] == x) || i %in% ignore){
    next
  }
  select_columns <- c(select_columns, i)
}

# perform PCA with scaling and center (avoid missleading eigenvectors)
all_together.pca <- prcomp(all_together[,select_columns], center=TRUE, scale.=TRUE )

groups <- as.factor(all_together$isFlaky)
levels(groups) <- c("non-flaky","flaky")
ggbiplot(all_together.pca, groups=groups, ellipse=TRUE, obs.scale=TRUE, var.scale=TRUE, alpha=0.1)
```



This plot with all data points is pretty useless since there are too many data points which overlap heavily! A way to tackle is to downsample the data set. Try to balance the data. Note that only 1.4% of the data are flaky.

```
nr_non_flaky <- 120
nr_flaky <- 40

non_flaky_sample <- dd.labeled[sample( which( !dd.labeled$isFlaky), nr_non_flaky), ]
flaky_sample <- dd.labeled[sample( which( dd.labeled$isFlaky), nr_flaky), ]

all_together.downsample <- rbind(non_flaky_sample, flaky_sample)


select_columns <- c()

# for-loop cannot handle data.table :-(
all_together.downsample <- as.data.frame(all_together.downsample)
# PCA cannot be performed if there are vectors if uniform entries
for (i in colnames(all_together.downsample)){
  x <- all_together.downsample[3,i]
  if(all(all_together.downsample[,i] == x) || i %in% ignore){
    next
  }
  select_columns <- c(select_columns, i)
}

# perform PCA with scaling and center (avoid missleading eigenvectors)
all_together.downsample.pca <- prcomp(all_together.downsample[,select_columns], center=TRUE, scale.=TRU

groups <- as.factor(all_together.downsample$isFlaky)
levels(groups) <- c("non-flaky","flaky")
ggbiplot(all_together.downsample.pca, groups=groups, ellipse=TRUE, obs.scale=TRUE, var.scale=TRUE)
```



A unique cluster of flaky tests cannot be identified if we consider all test of all projects together. I think one of the main insights of all the biplots so far is that flaky tests of a project build a cluster (in biplots) but only within a certain project. In the case where arbitrarily tests of different projects were chosen it is very unlikely to cluster them.

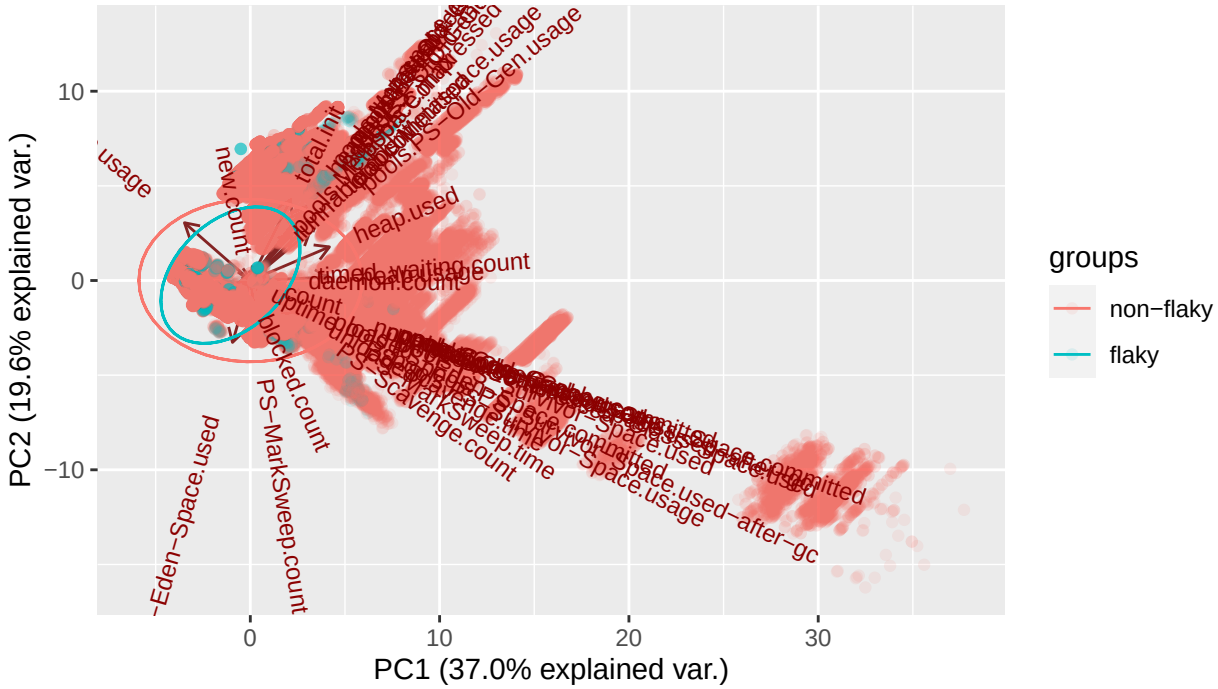
Reported flaky tests of the idFlaky data set

```
dd.idflakies <- merge( dd, idflakies, by.x=c('TestCase','CommitHash'), by.y=c('Test Name','SHA'), all=FA

nr_measure_idflakies_entries <- nrow( unique( dd.idflakies[, c('TestCase','CommitHash')]))
# proportion of measured idlfakies entries
nr_measure_idflakies_entries/nrow(idflakies)
```

```
## [1] 0.5379147
```

```
# label idflakies data set with isFlaky
dd.idflakies$isFlaky <- TRUE

# merege dd.idflakies with all non flaky tests
dd.labeled <- as.data.frame( dd.labeled)
dd.idflakies <- as.data.frame( dd.idflakies)
tmp <-  merge( dd.labeled[ which( dd.labeled$isFlaky == FALSE),], dd.idflakies[, c('TestCase','CommitHas
               by=c('TestCase','CommitHash'), all=TRUE, allow.cartesian=FALSE)

tmp <- unique( as.data.table(tmp))

tmp[isFlaky.y == TRUE, 'isFlaky.x'] <- TRUE
tmp$isFlaky.y <- NULL
tmp <- tmp[,1:74]
colnames(tmp)[length(colnames(tmp))] <- 'isFlaky'
```

```r
# data set of interest
idflakies_only <- na.omit(tmp)
rm(tmp)


select_columns <- c()

# for-loop cannot handle data.table :-(
idflakies_only <- as.data.frame(idflakies_only)
# PCA cannot be performed if there are vectors if uniform entries
for (i in colnames(idflakies_only)){
  x <- idflakies_only[3,i]
  if(all(idflakies_only[,i] == x) || i %in% ignore){
    next
  }
  select_columns <- c(select_columns, i)
}

# perform PCA with scaling and center (avoid missleading eigenvectors)
idflakies_only.pca <- prcomp(idflakies_only[,select_columns], center=TRUE, scale.=TRUE )

groups <- as.factor(idflakies_only$isFlaky)
levels(groups) <- c("non-flaky","flaky")
ggbiplot(idflakies_only.pca, groups=groups, ellipse=TRUE, obs.scale=TRUE, var.scale=TRUE, alpha=0.1)
```



Also in this case we can not clearly identrify a structure of flaky tests. It is similar like above where all measurements were considered. Let's differentiate again within certain projects.

Within projects with idFlakies flaky tests only

Below you see a list of the available measurements (not tests)

```
idflakies_only <- as.data.table( idflakies_only)
tests_only <- unique( idflakies_only[, c('ProjectName','CommitHash','isFlaky')])
idflakies_only[, list(flakyMeasurements=sum(isFlaky), notflakyMeasurements=sum(!isFlaky)),
               by=c('ProjectName','CommitHash')][,c('ProjectName','flakyMeasurements','notflakyMeasureme
```

```
##                     ProjectName flakyMeasurements notflakyMeasurements
##  1:                       c2mon                 0                  7432
##  2:                       orbit                20                  3700
##  3:               WebCollector                 20                   160
##  4:                     hutool                 20                 10140
##  5:                       oryx                 20                  7188
##  6:                 excelastic                 20                   200
##  7:                      esper                  0                    80
##  8:              rxjava2-extras                 20                  7680
##  9:                db-scheduler                 40                   920
## 10:                GoodData-CL                 20                   400
## 11:                    as2-lib                  0                   940
## 12:                        one                  0                   240
## 13: spring-cloud-zuul-ratelimit                 0                    60
## 14:                oci-java-sdk                 0                  2240
## 15:                    aletheia                60                   560
## 16:                    querydsl                 0                 25242
## 17:                        noxy                40                   540
## 18:                     helios                  0                  4034
## 19:                     admiral                 0                  2830
## 20:     delight-nashorn-sandbox                40                  1500
## 21:                aismessages                 40                   840
## 22:                         sos                 0                   100
## 23:            elastic-job-lite                200                 10960
## 24:           jhipster-registry                20                   610
## 25:                     sawmill                 0                    40
## 26:       vertx-rabbitmq-client                 20                    80
## 27:                         yawp                20                   420
## 28:                        cukes                20                  1060
## 29:    vertx-completable-future              1000                  2180
## 30:                     redpipe                100                  1120
## 31:                       whois                 0                 50800
## 32:                  marine-api                 40                 18460
## 33:      hsac-fitnesse-fixtures                 20                  5000
## 34:                    Activiti                 0                  3122
## 35:             incubator-dubbo                440                 35824
## 36:                      hadoop                128                  3392
## 37:               nifi-registry                 0                  5020
## 38:                      Struts                48                  4226
## 39:          fluent-logger-java                20                   300
## 40:                      tyrus                 20                 10140
## 41:             java-cas-client                20                  3000
## 42:              Java-WebSocket               1020                  1860
## 43:                     wildfly                860                  1322
## 44:                  jfreechart                20                 43500
```

```
## 45:       nexus-repository-helm                20                  340
## 46:            spring-cloud-aws                20                 5900
## 47:          spring-data-ebean                 20                  940
## 48:         spring-data-envers                 20                  180
## 49:                  spring-ws                  0                 2180
## 50:                    unix4j                  20                 8540
## 51:          wikidata-toolkit                 100                15620
## 52:       wildfly-maven-plugin                  0                  306
## 53:                    riptide                 40                28974
## 54:                   retrofit                180                11920
## 55:                      wro4j                 20                17424
## 56:                      pippo                100                 4220
## 57:                     timely                   6                  504
## 58:                       luwak                  0                 3360
##                   ProjectName flakyMeasurements notflakyMeasurements
```