Capstone Two Project: Genotyping SNP classification

Carsten Bruckner

https://github.com/cabruck/DataScienceCapstoneTwo

Overview

In the interest of time, using this document to capture additional data cleaning and exploratory data analysis steps, rather than implement all in Jupyter Notebook. This document is the second step of the analysis, where the first step was described in \git_repositories\DataScienceCapstoneTwo\reports\DataWranglingNotes.htm.

Input file:

- \git_repositories\DataScienceCapstoneTwo\data\data_cleaning_step1.zip\data_cleaning_step1.txt
    - This is the output from TIBCO Spotfire project \git_repositories\DataScienceCapstoneTwo\spotfire\data_cleaning_step1.dxp

The output file from this procedure:

- \git_repositories\DataScienceCapstoneTwo\data\data_cleaning_step1.zip\data_cleaning_step2.txt

Additional Data Cleaning Steps

From previous data review, it was discovered that there are placeholder values reported based not on data, but on static prior expectations. For example, if n_AA = 0 (no samples with AA genotype), might need to reset all AA.mean* metrics to Null, and same idea with n_AB=0, n_BB=0.

Additionally, the AB.meanX metric is known to have an optimum value near 0, and significant deviations in either direction usually portend performance issues. Therefore, for this metric only, also transform with its absolute value.

Therefore, I created 18 new columns with .clean column name suffix, replacing values based on no samples with "Null" value:

```
1. Select Data > Add data...
    Source: Data loaded from file
    Type: Text
    Location:
C:\Users\carsten.bruckner\OneDrive\Documents\Springboard\git_repositories\DataScienceCapstoneTwo\data\data_cl
eaning_step1.txt
    Data loaded at: 10/10/2021 10:23 AM
    Data was added as a new data table
2. Data > Add calculated column...
    Column name: AA.meanX.clean
    Expression: if(n_AA=0,Null,[AA.meanX])
3. Data > Add calculated column...
    Column name: AA.meanY.clean
    Expression: if(n_AA=0,Null,[AA.meanY])
4. Data > Add calculated column...
    Column name: AA.varX.clean
    Expression: if(n_AA=0,Null,[AA.varX])
5. Data > Add calculated column...
    Column name: AA.varY.clean
    Expression: if(n_AA=0,Null,[AA.varY])
6. Data > Add calculated column...
    Column name: AA.varX.Z.clean
    Expression: if(n_AA=0,Null,[AA.varX.Z])
7. Data > Add calculated column...
```

    Column name: AA.varY.Z.clean

    Expression: if(n_AA=0,Null,[AA.varY.Z])

8. Data > Add calculated column...

    Column name: AB.meanX.abs_clean

    Expression: if(n_AB=0,Null,Abs([AB.meanX]))

9. Data > Add calculated column...

    Column name: AB.meanY.clean

    Expression: if(n_AB=0,Null,[AB.meanY])

10. Data > Add calculated column...

    Column name: AB.varX.clean

    Expression: if(n_AB=0,Null,[AB.varX])

11. Data > Add calculated column...

    Column name: AB.varY.clean

    Expression: if(n_AB=0,Null,[AB.varY])

12. Data > Add calculated column...

    Column name: AB.varX.Z.clean

    Expression: if(n_AB=0,Null,[AB.varX.Z])

13. Data > Add calculated column...

    Column name: AB.varY.Z.clean

    Expression: if(n_AB=0,Null,[AB.varY.Z])

14. Data > Add calculated column...

    Column name: BB.meanX.clean

    Expression: if(n_BB=0,Null,[BB.meanX])

15. Data > Add calculated column...

    Column name: BB.meanY.clean

    Expression: if(n_BB=0,Null,[BB.meanY])

16. Data > Add calculated column...

    Column name: BB.varX.clean

    Expression: if(n_BB=0,Null,[BB.varX])

17. Data > Add calculated column...

    Column name: BB.varY.clean

    Expression: if(n_BB=0,Null,[BB.varY])

18. Data > Add calculated column...

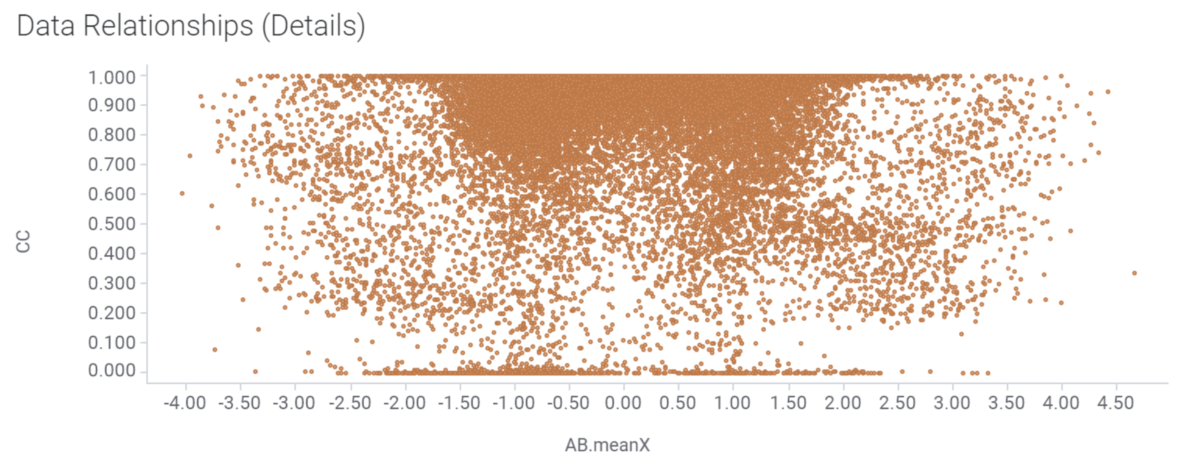
    Column name: BB.varX.Z.clean

    Expression: if(n_BB=0,Null,[BB.varX.Z])

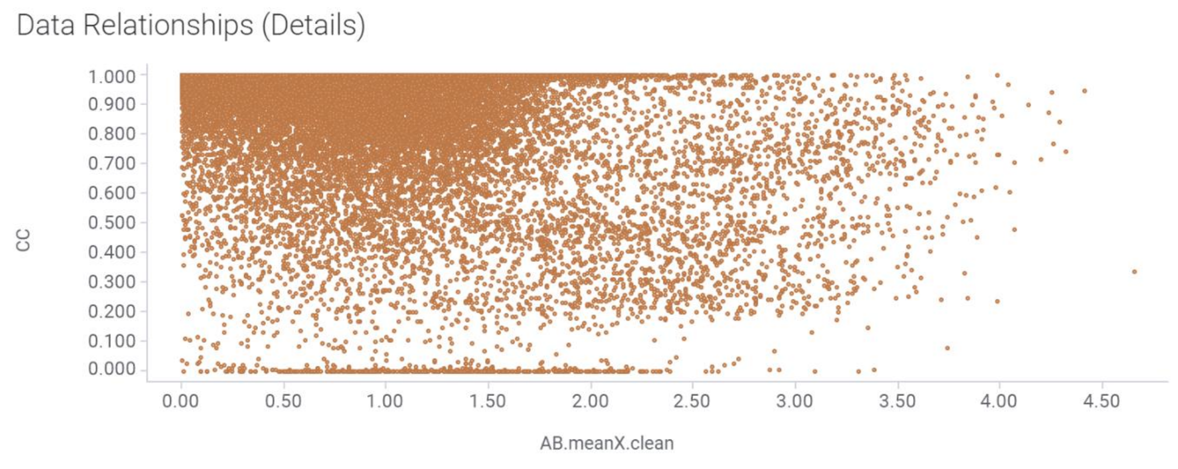
19. Data > Add calculated column...

    Column name: BB.varY.Z.clean

    Expression: if(n_BB=0,Null,[BB.varY.Z])

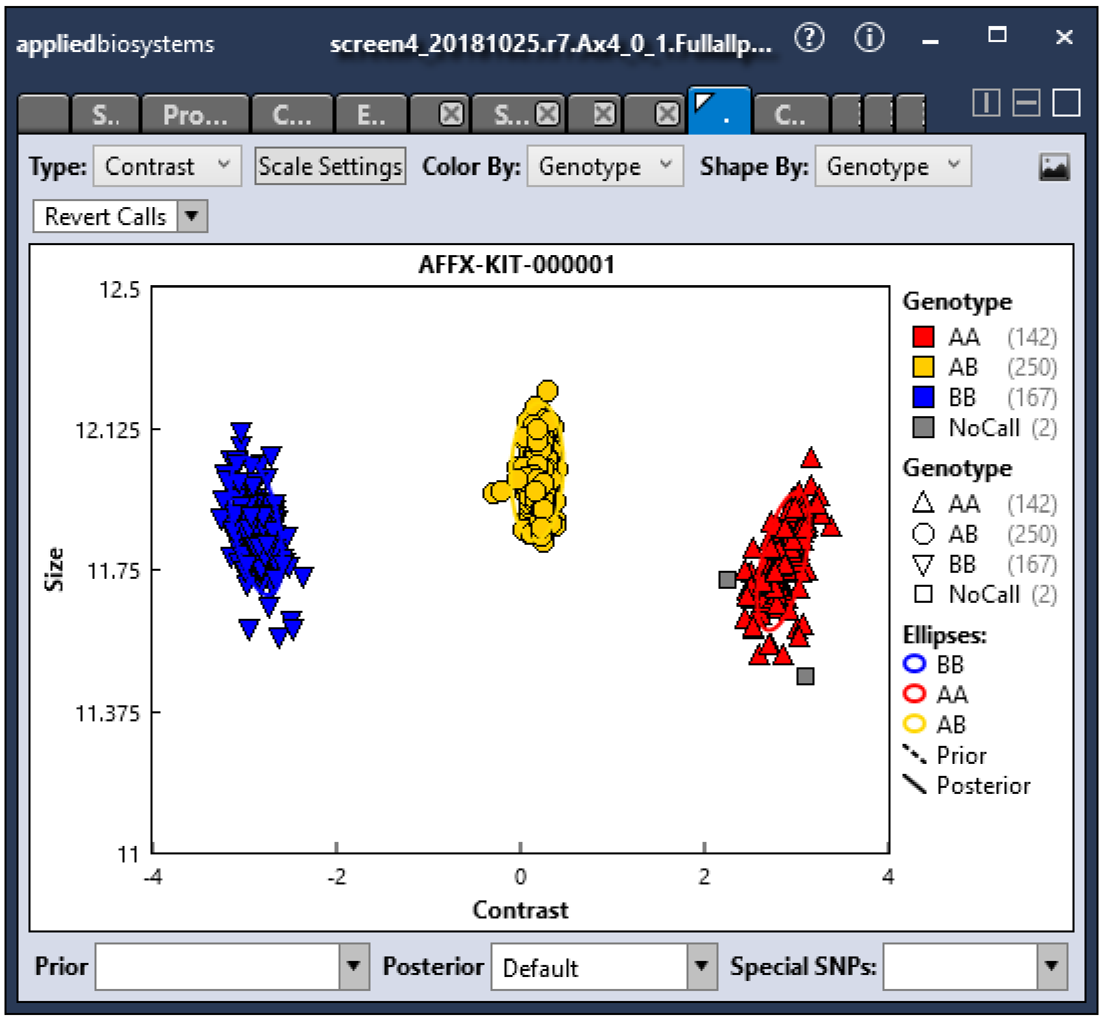


Following X axis metric later renamed to AB.mean.abs_clean



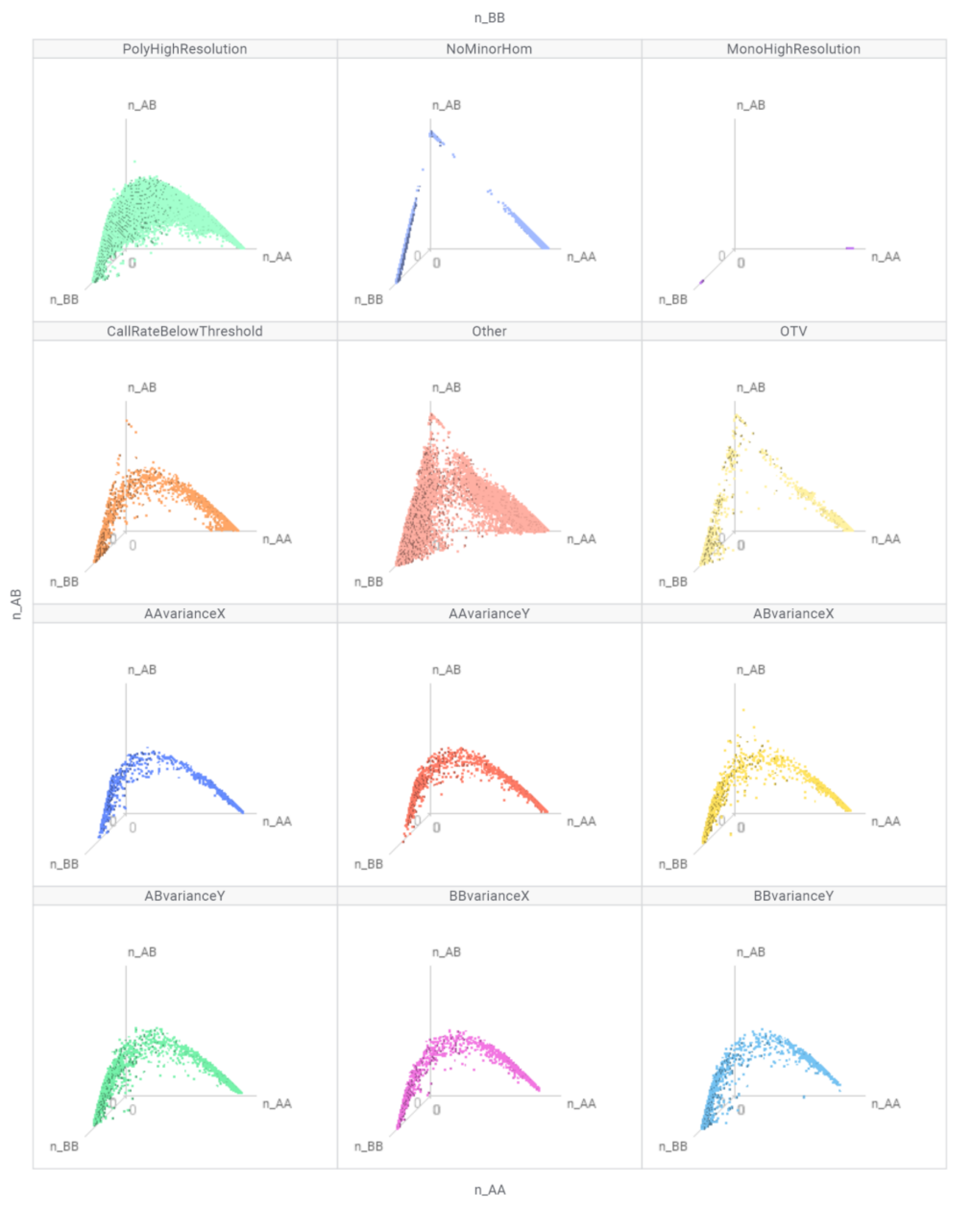
Sample count columns (those beginning with "n_") are dependent on the number of samples analysed, which will vary from day to day. The absolute value of the counts does not matter so much as the relative distribution of counts.

n_AA, n_AB, n_BB are each counts of tested DNA samples for current SNP (probeset). In example below, the counts appear in the legend. Some probesets will only have one genotype (only AA calls, so n_AB = 0, n_BB = 0). Some will have two reported genotypes (mix AA&AB, or AB&BB). An accurately-calling probesets won't have AA&BB but no AB, when sample set (n) is reasonably-sized.



If all the samples give a call, then n_AA + n_AB + n_BB = constant value. These statistics are strongly related. The following graph plots these three metrics against each other (3D scatter plot), grouped by different Conversion Types (pre-generated classification types). First row of plots are considered good quality classifications. Each data point is a different probeset. The entire visualization uses 808155 probesets.

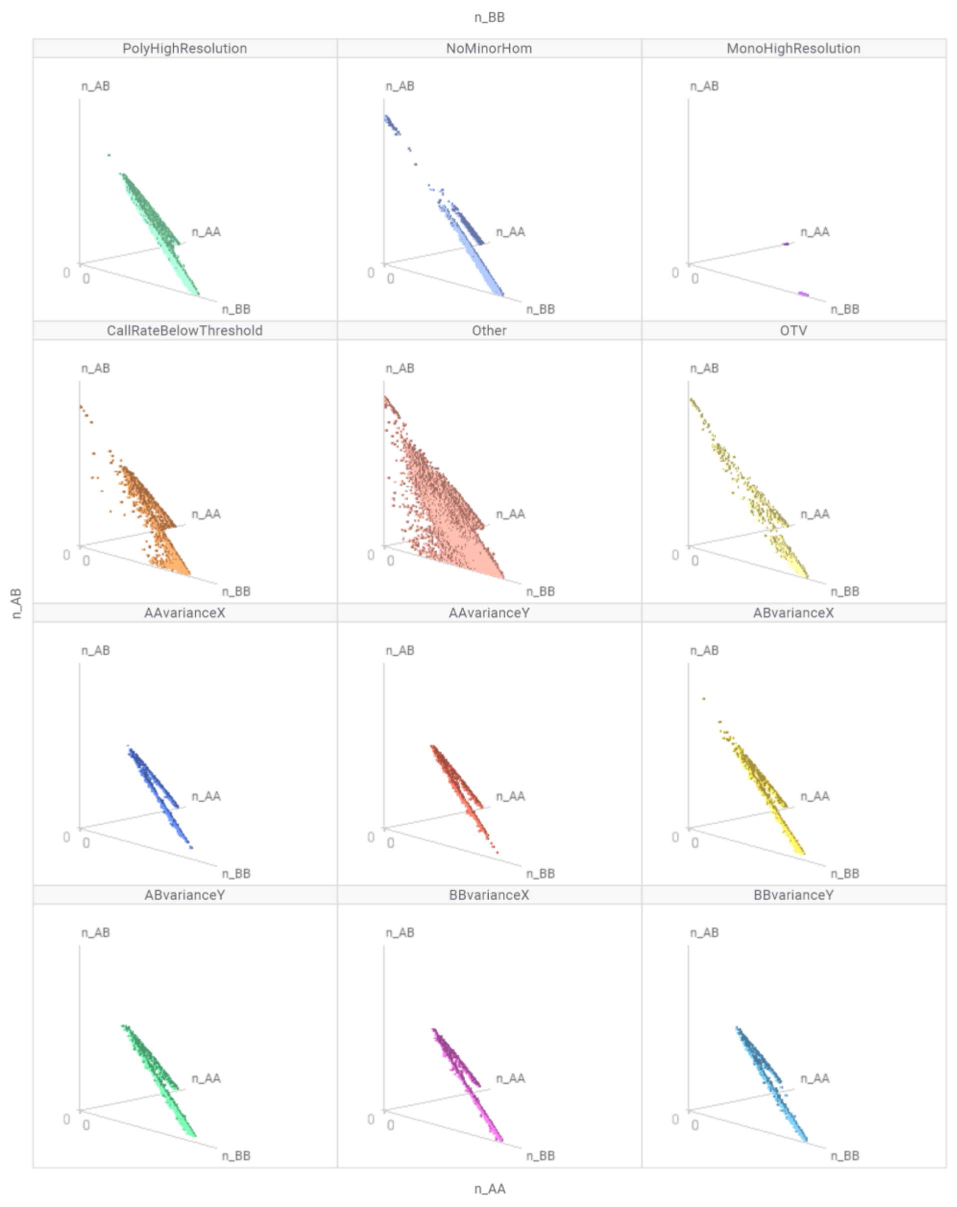
n_BB vs. n_AA and n_AB



Trellis by:
OriginalCT
Marker by:
(Row Number)
Color by:
OriginalCT

- 🟢 PolyHighResolution
- 🔵 NoMinorHom
- 🟣 MonoHighResolution
- 🟠 CallRateBelowThre...
- 🔴 Other
- 🟡 OTV
- 🔵 AAvarianceX
- 🔴 AAvarianceY
- 🟡 ABvarianceX
- 🟢 ABvarianceY
- 🟣 BBvarianceX
- 🔵 BBvarianceY

n_BB vs. n_AA and n_AB



Trellis by:
OriginalCT
Marker by:
(Row Number)
Color by:
OriginalCT

- 🟢 PolyHighResolution
- 🔵 NoMinorHom
- 🟣 MonoHighResolution
- 🟠 CallRateBelowThre...
- 🔴 Other
- 🟡 OTV
- 🔵 AAvarianceX
- 🔴 AAvarianceY
- 🟡 ABvarianceX
- 🟢 ABvarianceY
- 🟣 BBvarianceX
- 🔵 BBvarianceY

Outliers from the typical arch in PolyHighResolution or NoMinorHom patterns are more likely to have quality issues. Interestingly, there's a cluster of "NoMinorHom" samples with high n_AB but very low counts of n_AA and n_BB. This is a very unusual genotype distribution, suggesting these probesets are underperforming. These probesets should hopefully be detected as problematic by a good prediction model of quality.

The symmetry of the counts, in conjunction for there being no physical basis why low n_AA counts should be categorized any differently than low n_BB counts, suggests that these three variables can be transformed into one or two simpler metrics, and also convert raw counts into more data-set independent fractions:

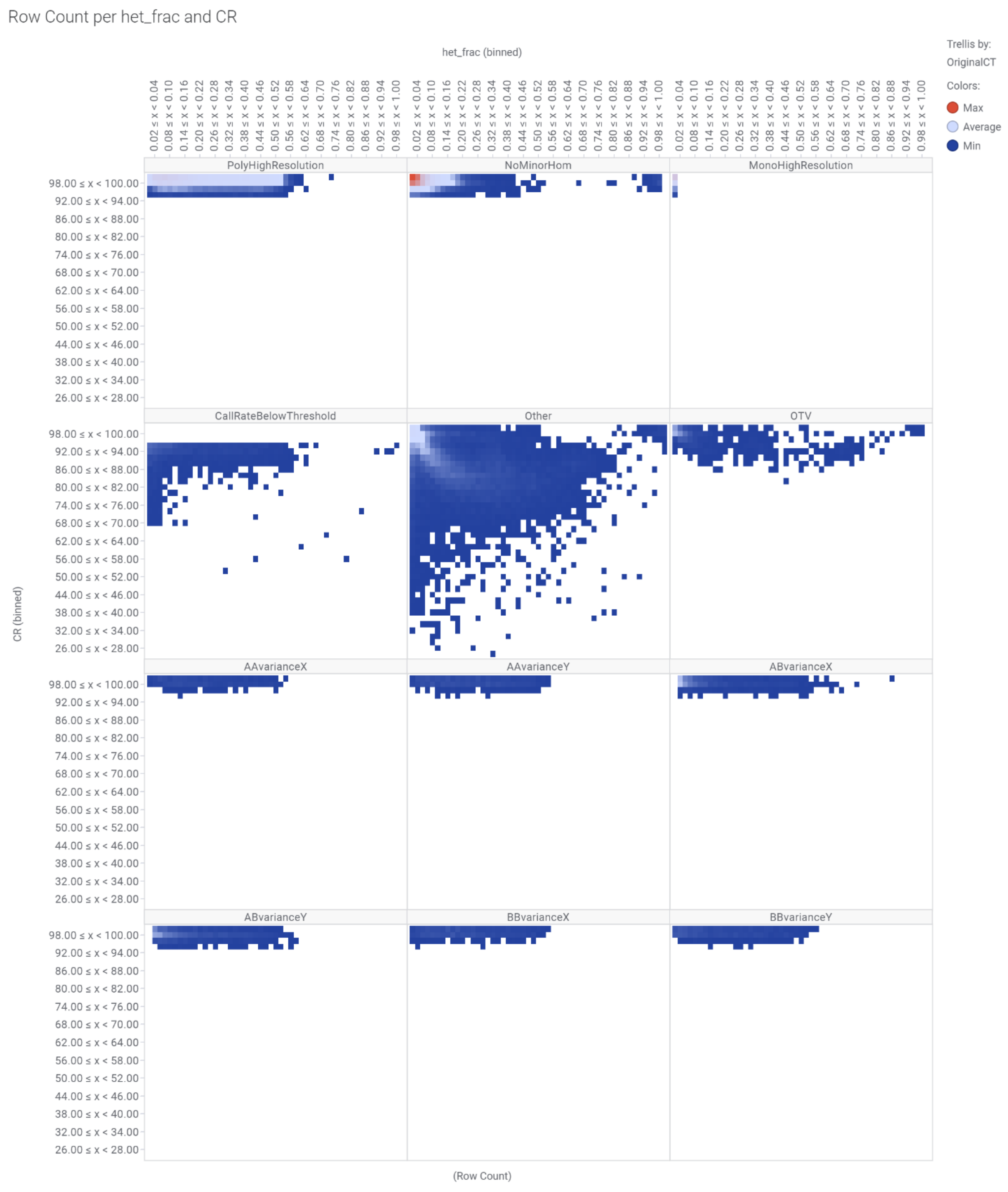
hom_frac = (n_AA + n_BB)/(n_AA + n_AB + n_BB)

het_frac = n_AB/(n_AA + n_AB + n_BB)

Since hom_frac can be derived using only het_frac, only het_frac can be used. Since this metric ignores NoCalls (n_NC), the independent metric Call Rate (CR) is still useful.

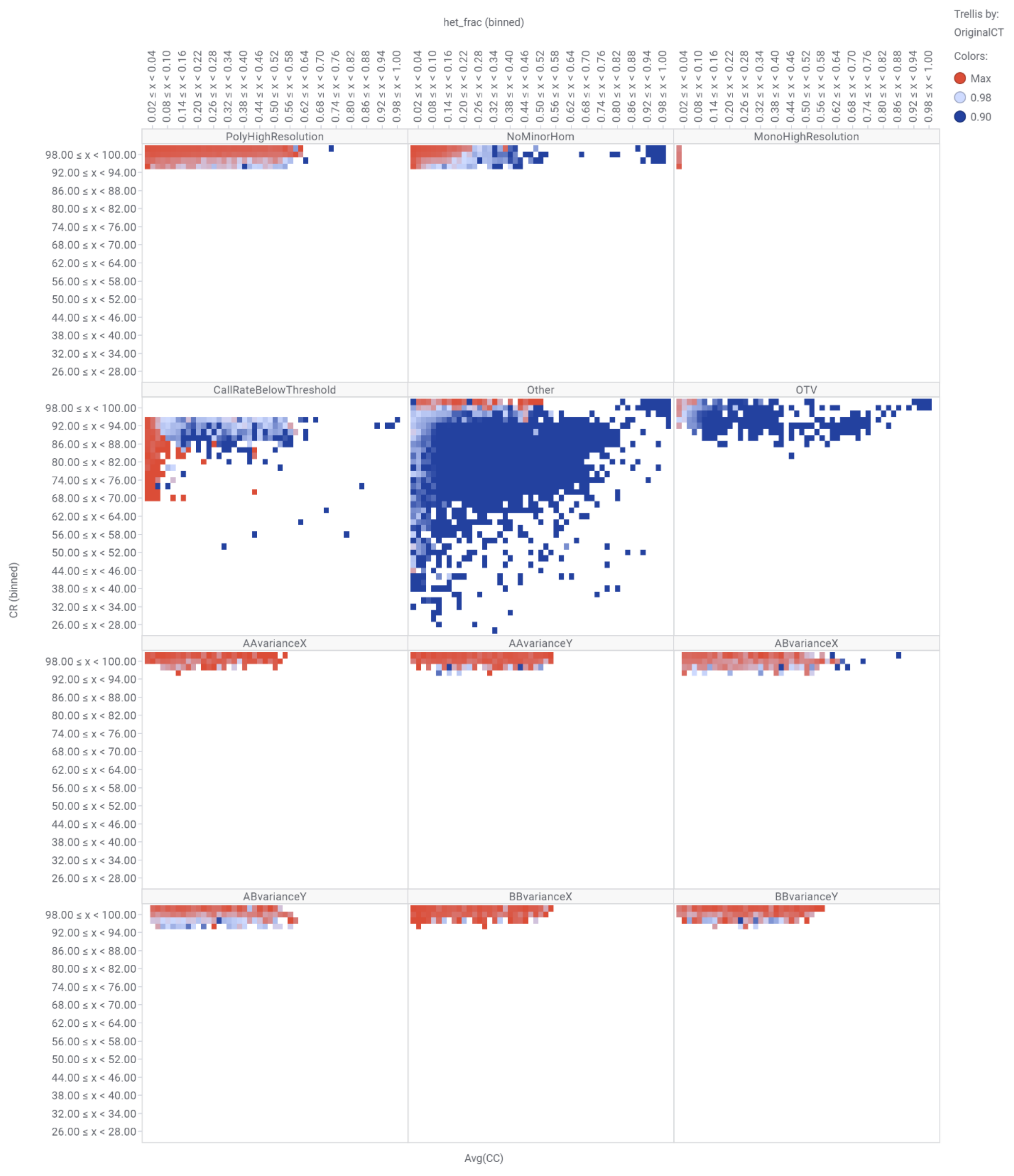
```
20. Data > Add calculated column...
    Column name: het_frac
    Expression: [n_AB] / Sum([n_AA],[n_AB],[n_BB])
```

The following plot is binned call rate vs het_frac, colored by the number of probesets in each bin (red most common), and grouped by Conversion Type ("OriginalCT"). Generally good probesets have a call rate >95%, and a het_frac < 0.6. The first row of plots are supposedly good conversion types, but there is a group of NoMinorHom probesets with het_frac > 0.6 that appear as outliers.



Now let's color the same plot by a metric we want to predict but usually can't measure, "concordance (CC)". In this heat map, the color is by average CC of probesets in each bin. Concordance values below 0.98 are problematic.

CC per het_frac and CR



A couple things immediately stand out:

- NoMinorHom, a supposedly-good category, has mostly problematic probesets when het_frac > 0.25. The exact het_frac threshold in this category depends n_AA+n_AB+n_BB.
- The 6 “variance” conversion types (drawn from probesets otherwise considered PolyHighResolution) generally include good probesets, and probably should also be considered good conversion types.
- The “Other” OriginalCT (a supposedly-bad category) still includes some good probesets (those with >95% call rate and good (red) concordance).

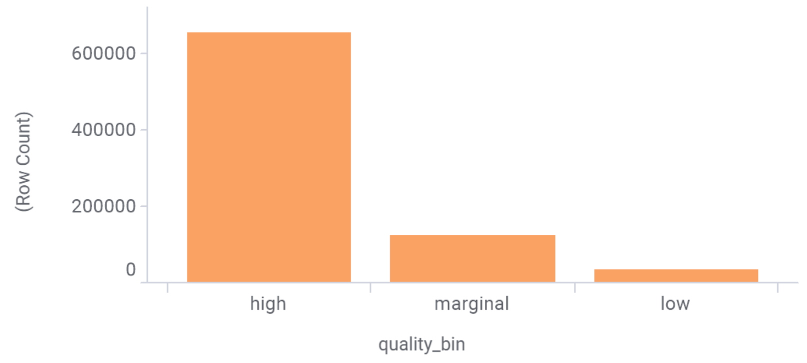
This plot suggests that perhaps the most simplistic model that might be pretty good at discriminating “high CR and CC” from remainder is:

[All with CR > 95% AND het_frac <0.6], less [NoMinorHom AND het_frac> 0.25], less [OriginalCT = “OTV”]

But bad probesets might be hidden in each OriginalCT category. Let’s create the “quality_bin” categorical response variable, which requires both call rate and concordance to be high. Then and see what substandard probesets might be hiding in the MonoHighResolution conversion type.

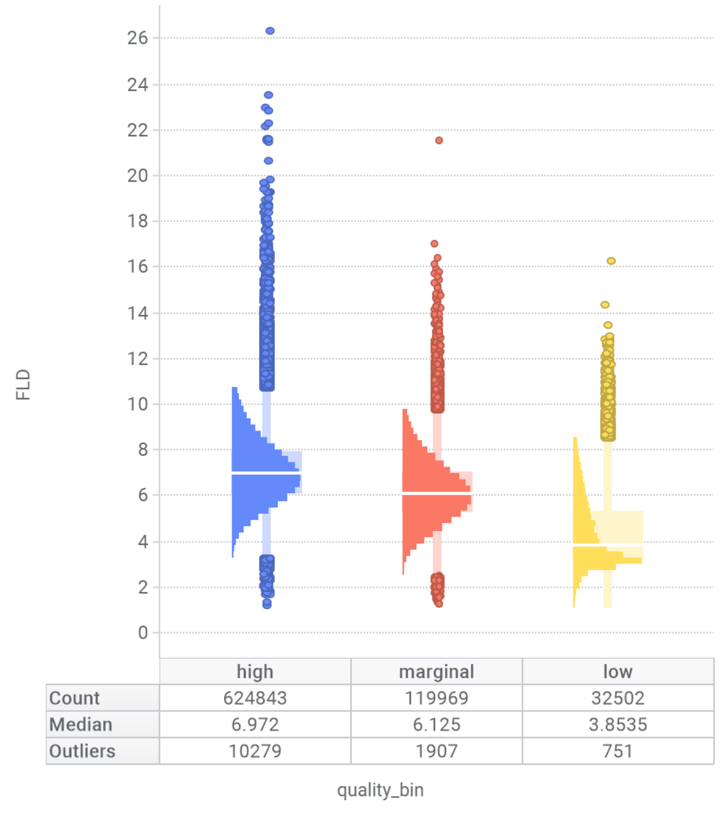
```
21. Data > Add calculated column...
    Column name: quality_bin
    Expression: Case when ([CC]>=0.995) and ([CR]>=98.5) then "high" –
when ([CC]>=0.985) and ([CR]>=95) then "marginal"
else "low"
end
```

Distribution – quality_bin



A measure of probeset quality is the variance-scaled distance between genotype clusters, “FLD” (Fisher’s linear discriminant). As expected, probesets with a higher FLD are more likely to belong to the “high” quality_bin:

Box plot



	high	marginal	low
Count	624843	119969	32502
Median	6.972	6.125	3.8535
Outliers	10279	1907	751

In the following table, 3% of MonoHighResolution probesets are considered “low quality”. This highlights a need to try to use some additional metrics to label at least these 3% of probesets.

		Stats by quality_bin					
		count of probesets			% of probesets		
OriginalCT.recommended	OriginalCT	high	marginal	low	high	marginal	low
TRUE	PolyHighResolution	191118	37684	6774	81%	16%	3%
TRUE	NoMinorHom	423812	74654	5736	84%	15%	1%
TRUE	MonoHighResolution	25750	2116	952	89%	7%	3%
FALSE	CallRateBelowThreshold			2082	0%	0%	100%
FALSE	Other	3795	4382	16754	15%	18%	67%
FALSE	OTV	318	610	1026	16%	31%	53%
FALSE	AAvarianceX	671	141	74	76%	16%	8%
FALSE	AAvarianceY	1125	260	73	77%	18%	5%
FALSE	ABvarianceX	1244	1007	256	50%	40%	10%
FALSE	ABvarianceY	1263	1061	549	44%	37%	19%
FALSE	BBvarianceX	1094	265	67	77%	19%	5%
FALSE	BBvarianceY	981	319	142	68%	22%	10%

It looks like a significant fraction of the “variance” OriginalCT have “high” quality probesets that could be recovered. Perhaps only the ABvarianceX and AbvarianceY categorical values might be used to reject probesets, since less than 50% of these probesets are high quality.

In general, before doing any model building, the baseline performance of labeling probesets as “low” quality vs “not low” quality using ONLY OriginalCT can be seen in the confusion matrix:

Confusion matrix	count of probesets by quality_bin

OriginalCT.recommended	high	marginal	low
TRUE	640680	114454	13462
FALSE	10491	8045	21023

correctly classified using only OriginalCT	96.0%
% high and marginal quality probesets recommended	97.6%
% low quality probesets not recommended	61.0%

Perhaps we can create a model that labels a larger percentage of the quality_bin = “low” probesets, and also identifies more of the “high” and “marginal” probesets, than we currently do using only OriginalCT categories.

How about deriving a continuous reponse variable, if we want to investigate models that do regression instead of pure classification?

```
22. Data > Add calculated column...
    Column name: quality_score
    Expression: Max((((4 * [CC] * 100) + (1 * [CR])) / 5) - 95,0)
```

The new “quality_score” variable ranges from 0-5. Concordance is 4x as important as Call Rate.

95% CC and 95% CR has quality score 0

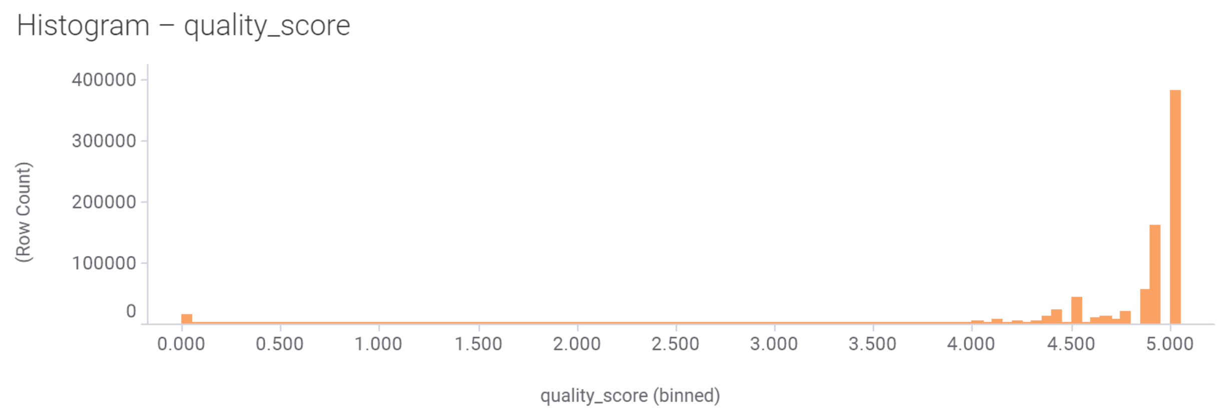
96% CC and 96% CR has quality score 1

95% CC and 100% CR has quality score 1

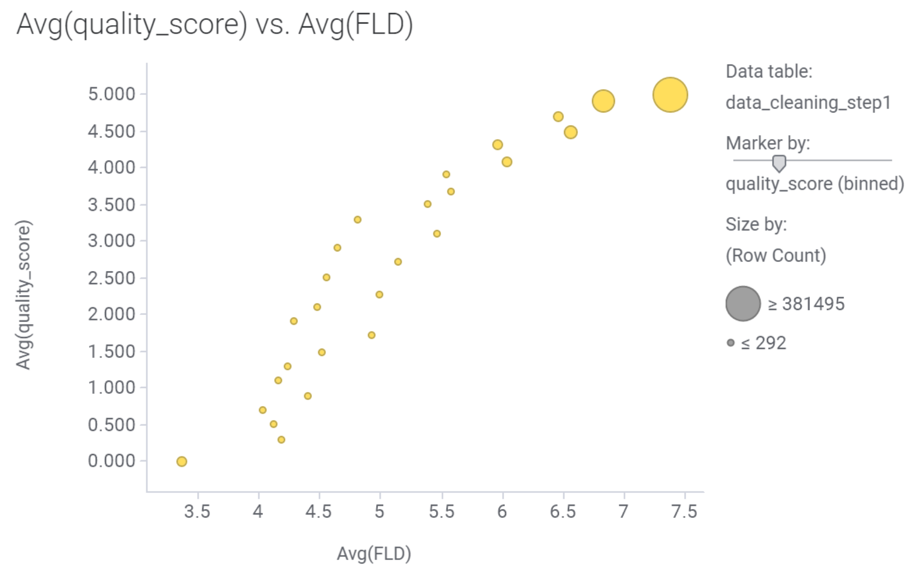
98% CC and 98% CR has quality score 3

100% CC and 100% CR has quality score 5

Most probesets have a quality score of at least 4.5:



Quality score is correlated with FLD (a measure of cluster separation):



Correlation of each variable vs continuous quality_score

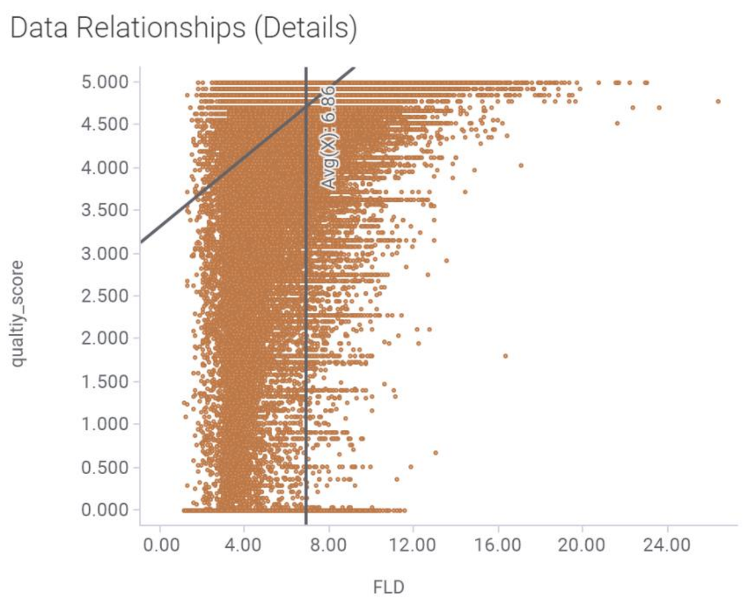
Y (numerical)	X (numerical)	p-value	FStat	RSq	R	Df	n
quality_score	CR	0.00E+00	1010039	0.56	0.75	808153	808155
quality_score	CC	0.00E+00	601603	0.43	0.65	808153	808155
quality_score	CC_het	0.00E+00	452886	0.37	0.61	764298	764300
quality_score	FLD	0.00E+00	163322	0.17	0.42	777312	777314

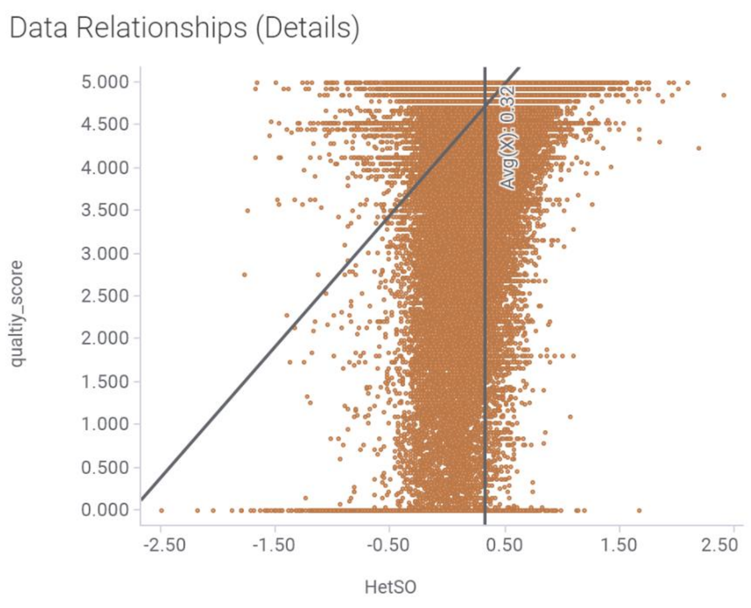
quality_score	HetSO	0.00E+00	134379	0.15	0.38	777341	777343
quality_score	AB.meanY.clean	0.00E+00	71362	0.08	0.29	777341	777343
quality_score	AB.varY.clean	0.00E+00	66862	0.08	-0.28	777341	777343
quality_score	HomRO	0.00E+00	59725	0.07	0.26	808124	808126
quality_score	AB.meanX.abs_clean	0.00E+00	56757	0.07	-0.26	777341	777343
quality_score	HomFLD	0.00E+00	50359	0.16	0.40	258879	258881
quality_score	AA.meanX.clean	0.00E+00	42568	0.08	0.29	467349	467351
quality_score	het_frac	0.00E+00	42237	0.05	-0.22	808153	808155
quality_score	AB.varX.clean	0.00E+00	40458	0.05	-0.22	777341	777343
quality_score	MinorAlleleFrequency	0.00E+00	27436	0.03	-0.18	808153	808155
quality_score	H.W.p-Value	0.00E+00	24296	0.03	0.17	808153	808155
quality_score	meanY	0.00E+00	19114	0.02	0.15	808153	808155
quality_score	BB.meanX.clean	0.00E+00	15456	0.03	-0.16	599654	599656
quality_score	AB.varX.Z.clean	0.00E+00	10957	0.04	-0.21	246166	246168
quality_score	HomHet	0.00E+00	9669	0.01	0.11	808153	808155
quality_score	MMD	0.00E+00	9215	0.03	0.19	258514	258516
quality_score	Hom.meanY.delta	0.00E+00	9164	0.01	-0.11	808153	808155
quality_score	AB.varY.Z.clean	0.00E+00	8972	0.04	-0.19	246166	246168
quality_score	BB.meanY.clean	0.00E+00	8737	0.01	0.12	599654	599656
quality_score	AA.meanY.clean	0.00E+00	7950	0.02	0.13	467349	467351
quality_score	Nclus	0.00E+00	7950	0.01	-0.10	808153	808155
quality_score	BB.varX.clean	0.00E+00	1629	0.00	0.05	599654	599656
quality_score	AA.varY.Z.clean	2.19E-266	1219	0.00	-0.07	246166	246168
quality_score	BB.varY.Z.clean	3.79E-231	1056	0.00	-0.07	246166	246168
quality_score	AA.varX.Z.clean	1.95E-219	1002	0.00	-0.06	246166	246168
quality_score	BB.varX.Z.clean	4.65E-218	996	0.00	-0.06	246166	246168
quality_score	AA.varX.clean	1.22E-134	610	0.00	0.04	467349	467351
quality_score	AA.varY.clean	5.15E-33	143	0.00	-0.02	467349	467351
quality_score	BB.varY.clean	1.12E-29	128	0.00	0.01	599654	599656

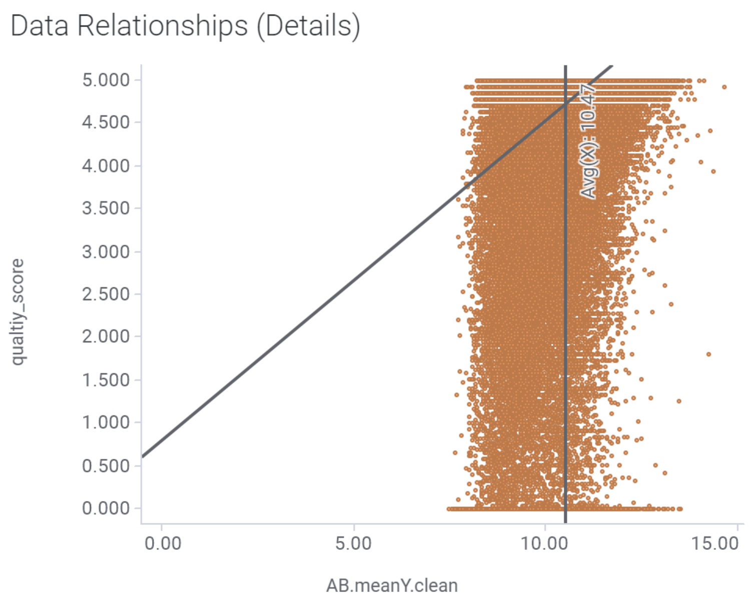
CR and CC are used to compute quality score, and CC_het is highly correlated to CC. Normally CC and CC_het are not available, and so shouldn't be used in any useful model.

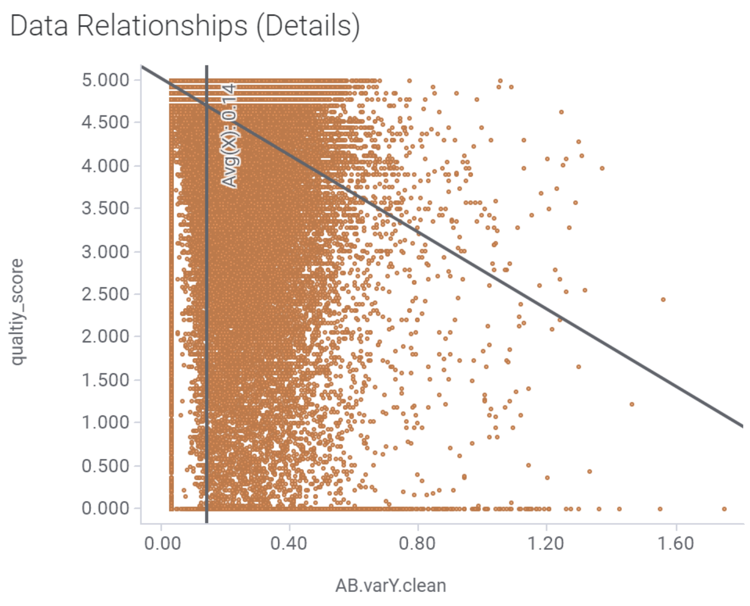
The sign of the Pearson correlation (R) is generally consistent with the metrics' relationships to predicted quality.

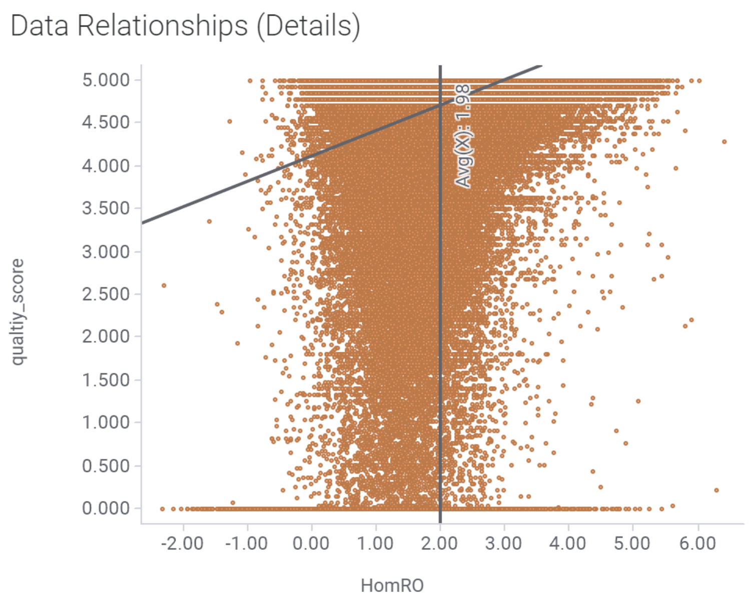
Some plots of the most correlated variables to quality_score, with best fit straight line (most probesets have a quality score of at least 4.5):

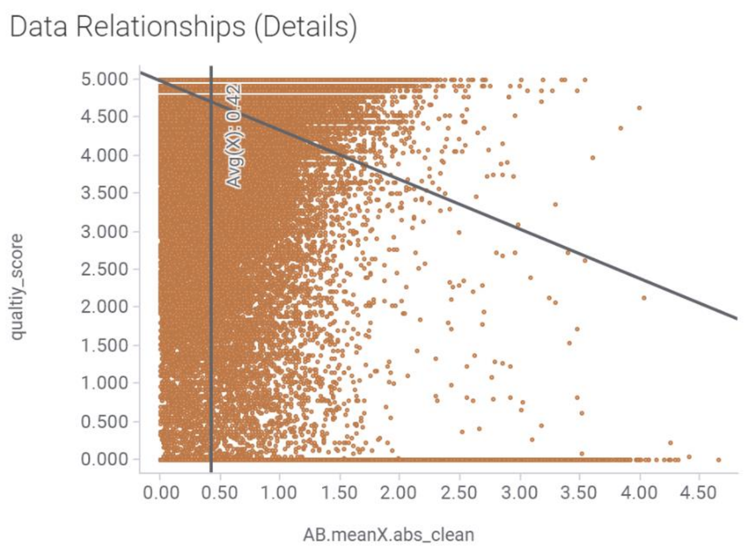






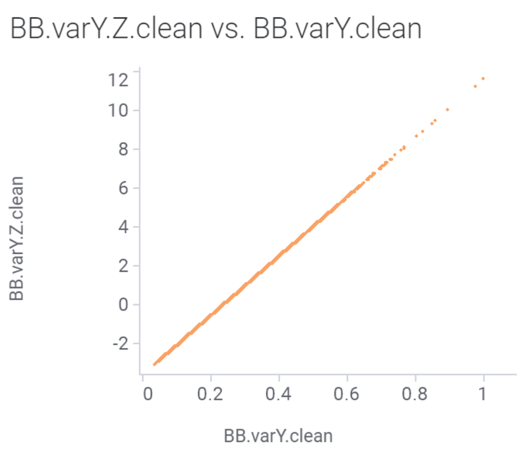


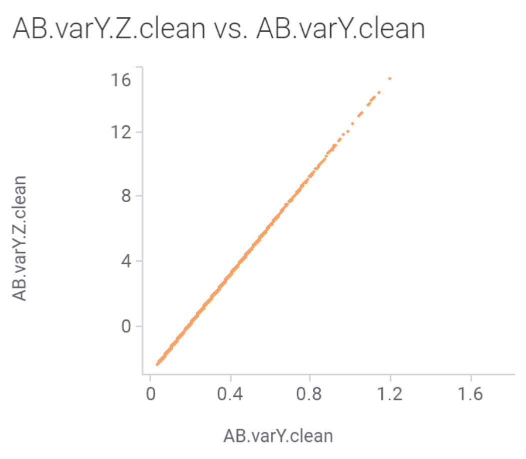


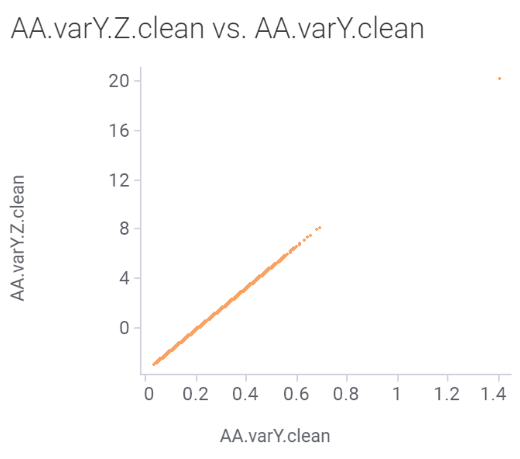


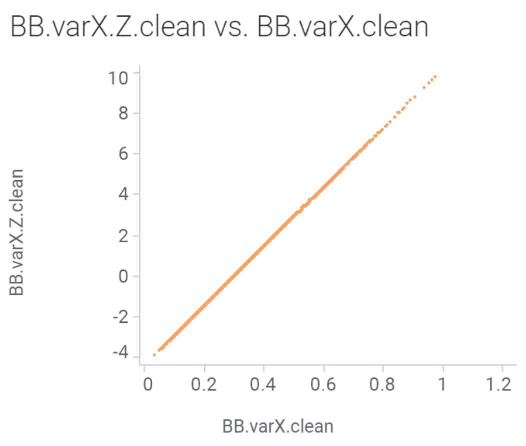
Variance terms have a floor at 0.03 (see AB.varY.clean example).

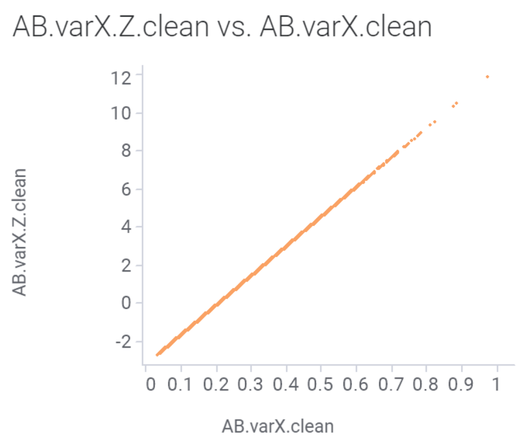
A number of z-score rescaled metrics have perfect correlation with the original metrics (see figure below). However, the z-score versions are computed on fewer probesets. From the previous correlation table vs quality_score, the sparser z-score versions are less strongly correlated than the untransformed versions. So let's omit the *.Z.clean variables from further analysis.

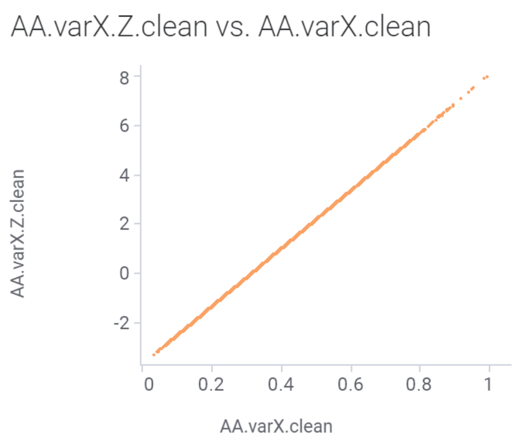






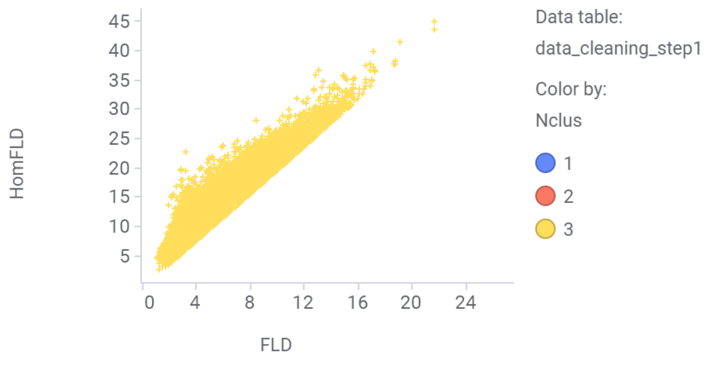






FLD and HomFLD are highly correlated, and FLD is available for more observations. Drop HomFLD from subsequent analysis.

HomFLD vs. FLD

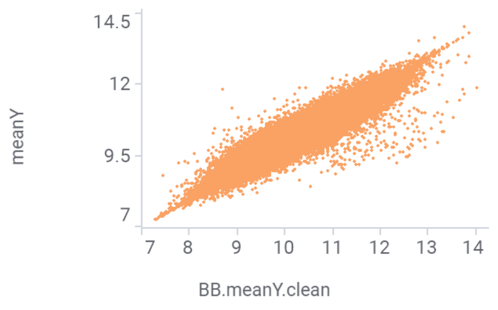


Nclus	(Row Count)	Count(FLD)	Count(HomFLD)
1	30476	0	0
2	519163	518798	365
3	258516	258516	258516

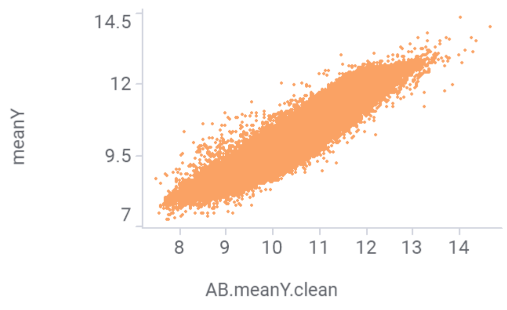
The signal strengths (meanY) of the three clusters AA, AB, BB are highly correlated, as expected. I made a weighted average metric "meanY" to see if a consolidated metric might perform better in modeling. "AB.meanY.clean" has the highest F-stat and R2 to quality_score, although that metric is not available for all probesets, unlike "meanY".

Probably only up to one of these 4 metrics will be used in final model.

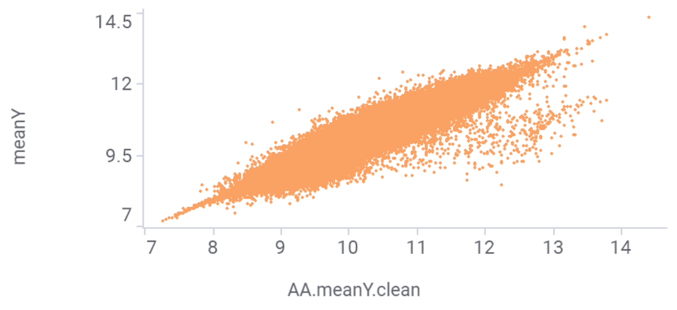
meanY vs. BB.meanY.clean



meanY vs. AB.meanY.clean

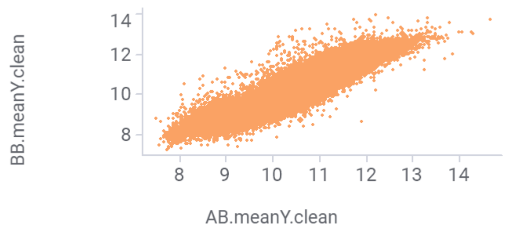


meanY vs. AA.meanY.clean

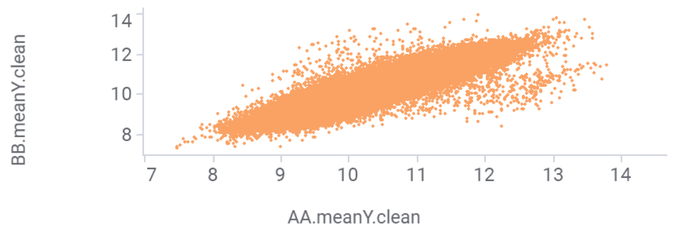


Text area

BB.meanY.clean vs. AB.meanY.clean



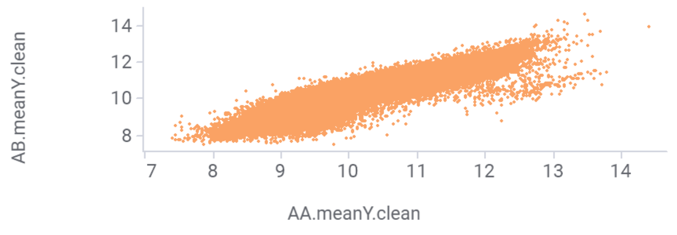
BB.meanY.clean vs. AA.meanY.clean



Text area

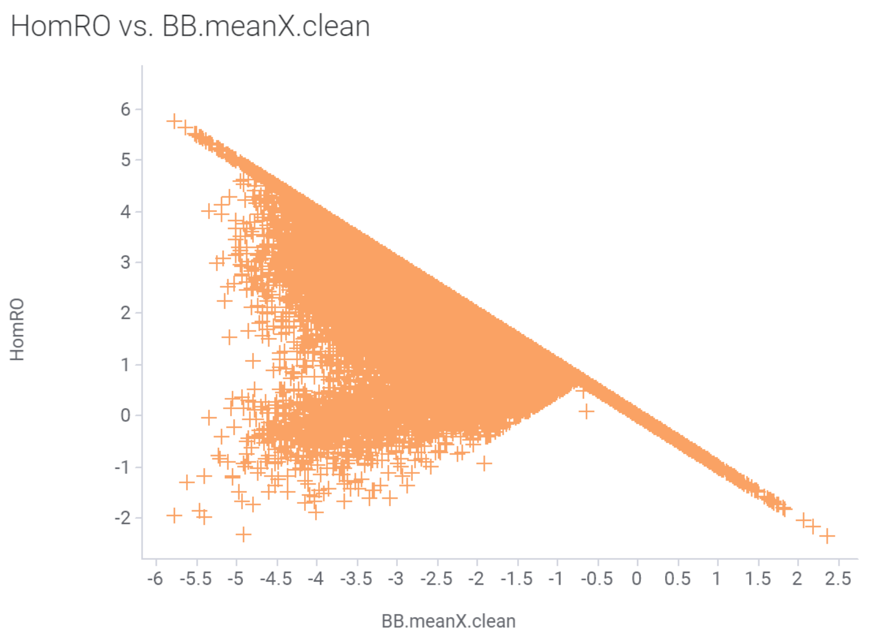
Text area

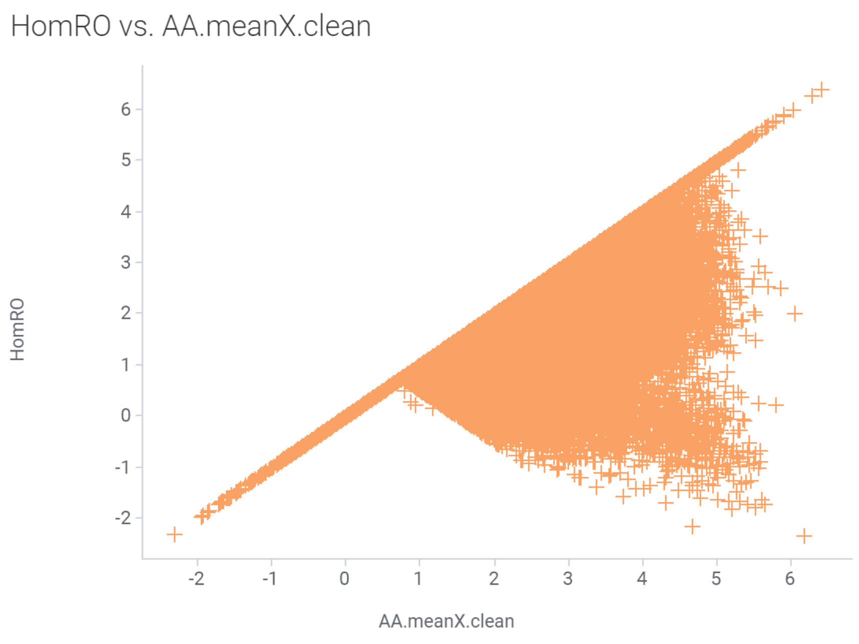
AB.meanY.clean vs. AA.meanY.clean



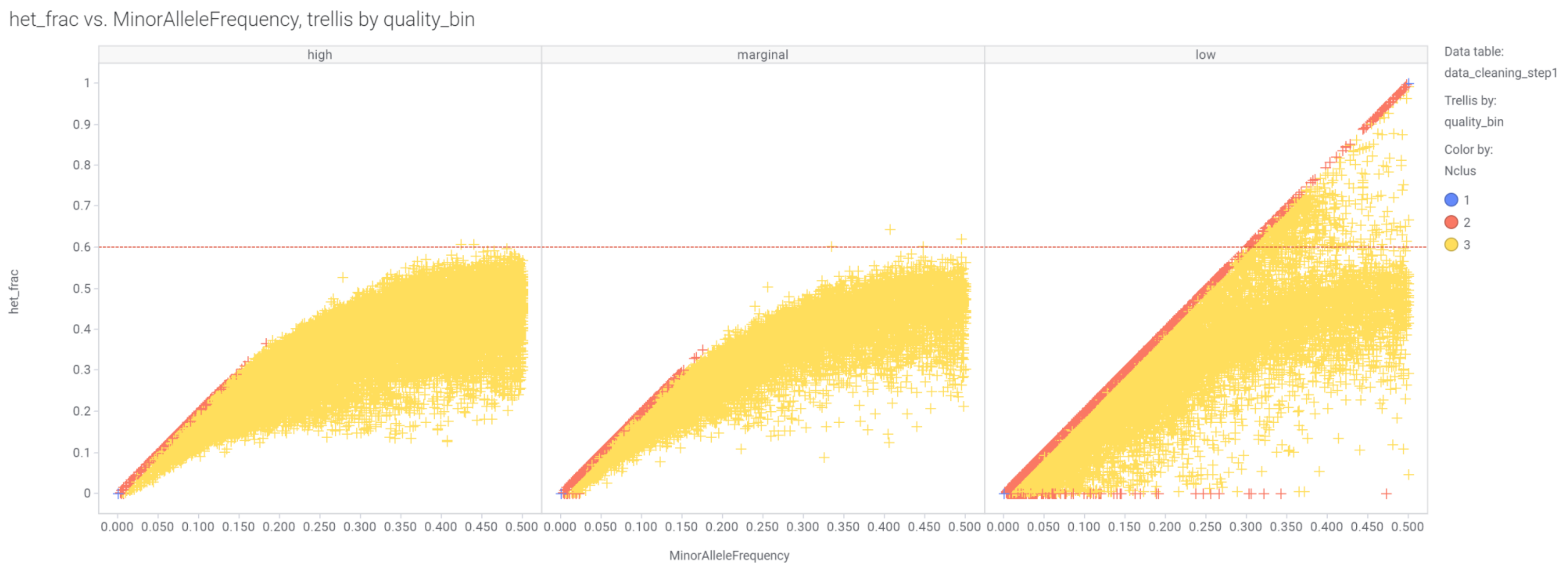
The AA or BB cluster with the minimum offset from contrast = 0 has its meanX used to compute HomRO. So BB.meanX.clean and AA.meanX.clean are highly correlated to derived HomRO.

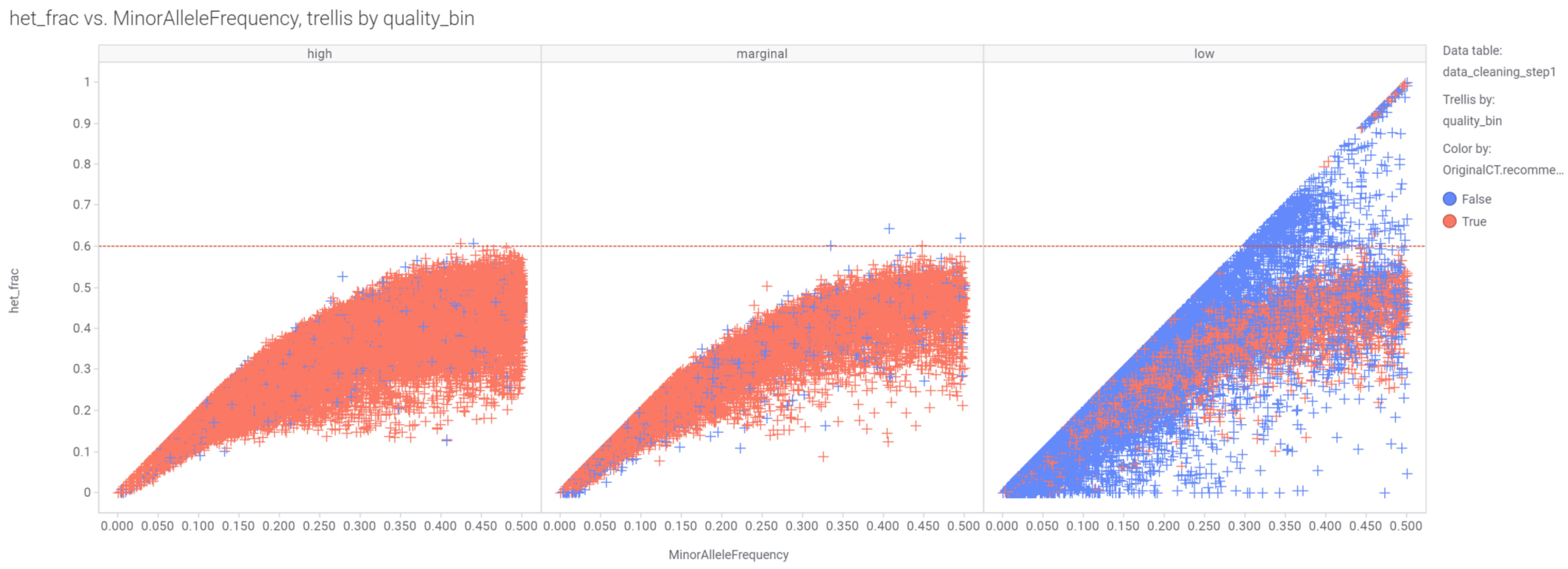
Probably only [HomRO] or [BB.meanX.clean AND AA.meanX.clean] may be use in final model, not both.





Het_frac and MinorAlleleFrequency are interrelated, because both are based on the count of samples in each genotype cluster (see two figures below). When Nclus=1, both are 0. When Nclus=2, they are perfectly correlated. One advantage of het_frac is that it is good at identifying low quality probesets when het_frac > 0.6, and also when Nclus = 2 AND het_frac = 0.





Pairwise correlation of remaining continuous variables

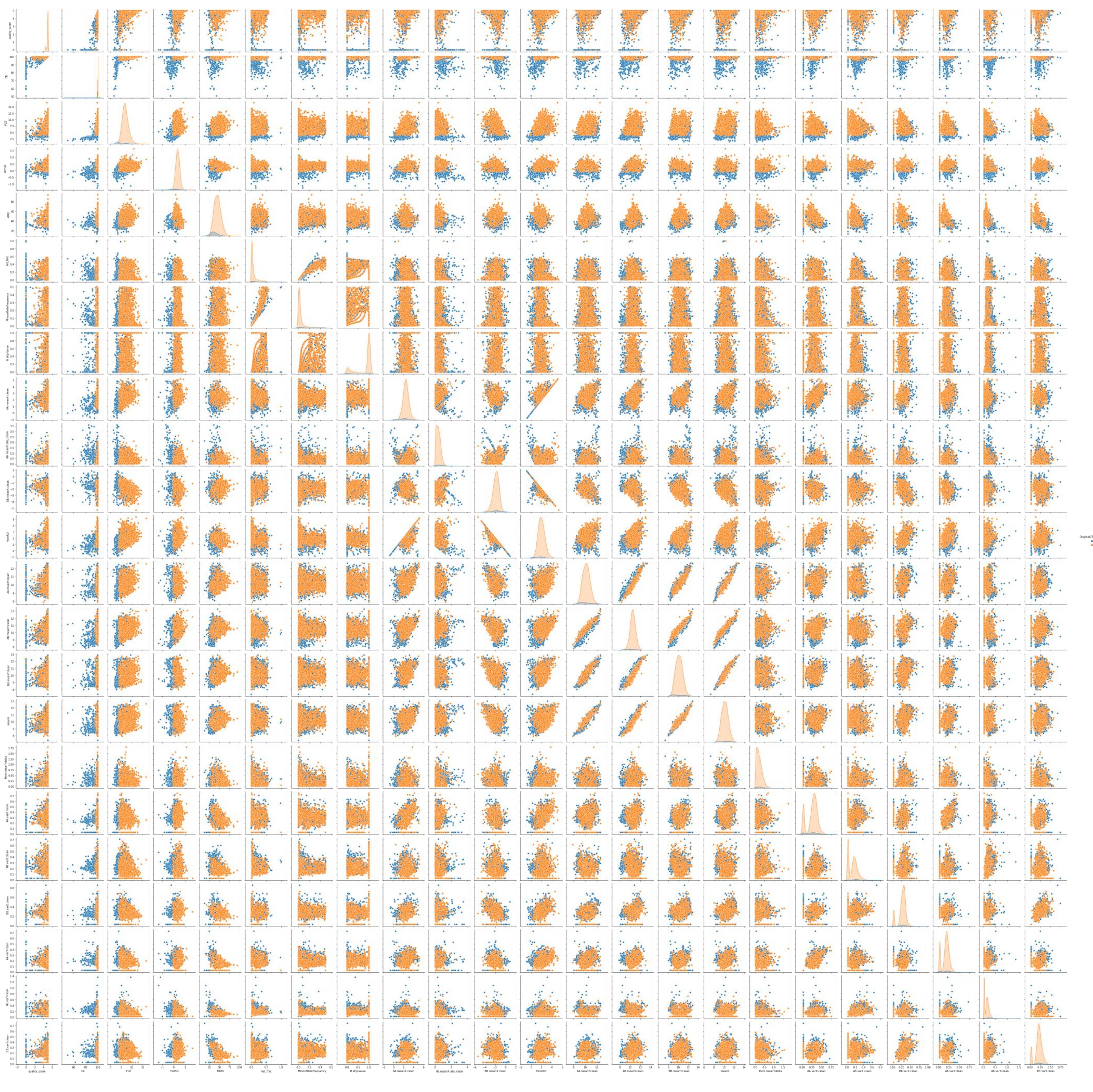
The following plot was made using

/git_repositories/DataScienceCapstoneTwo/notebooks/EDA_part2.ipynb

cols_plot = `['probeset_id', 'quality_bin', 'quality_score', 'OriginalCT.recommended',`

```
        'OriginalCT', 'CR', 'FLD', 'HetSO', 'MMD', 'het_frac',
        'MinorAlleleFrequency', 'H.W.p-Value', 'AA.meanX.clean',
        'AB.meanX.abs_clean', 'BB.meanX.clean', 'HomRO', 'AA.meanY.clean',
        'AB.meanY.clean', 'BB.meanY.clean', 'meanY', 'Hom.meanY.delta',
        'AA.varX.clean', 'AB.varX.clean', 'BB.varX.clean', 'AA.varY.clean',
        'AB.varY.clean', 'BB.varY.clean']
```

```
_ = sns.pairplot(ps_data[cols_plot].sample(n=10000),diag_kind='kde',hue='OriginalCT.recommended')
```


quality_score
CR
FLD
HetSO
MMD
het_frac
MinorAlleleFrequency
H.W.p-Value
AA.meanX.clean
AB.meanX.abs_clean
BB.meanX.clean
HomRO
AA.meanY.clean
AB.meanY.clean
BB.meanY.clean
meanY
Hom.meanY.delta
AA.varX.clean
AB.varX.clean
BB.varX.clean
AA.varY.clean
AB.varY.clean
BB.varY.clean