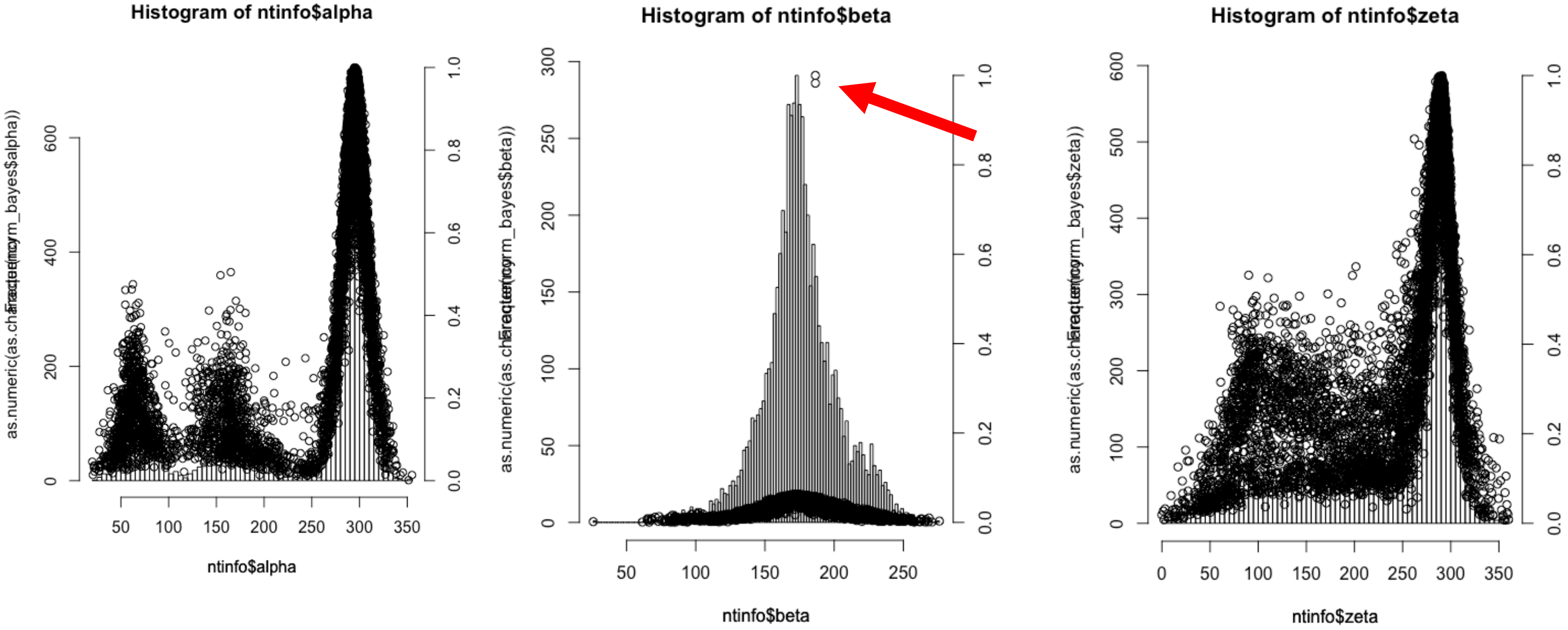
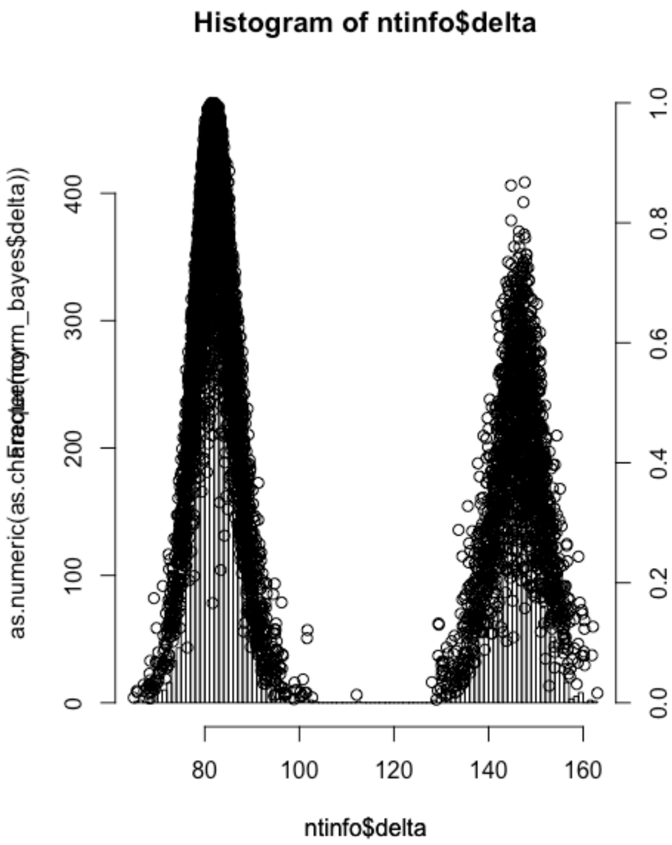
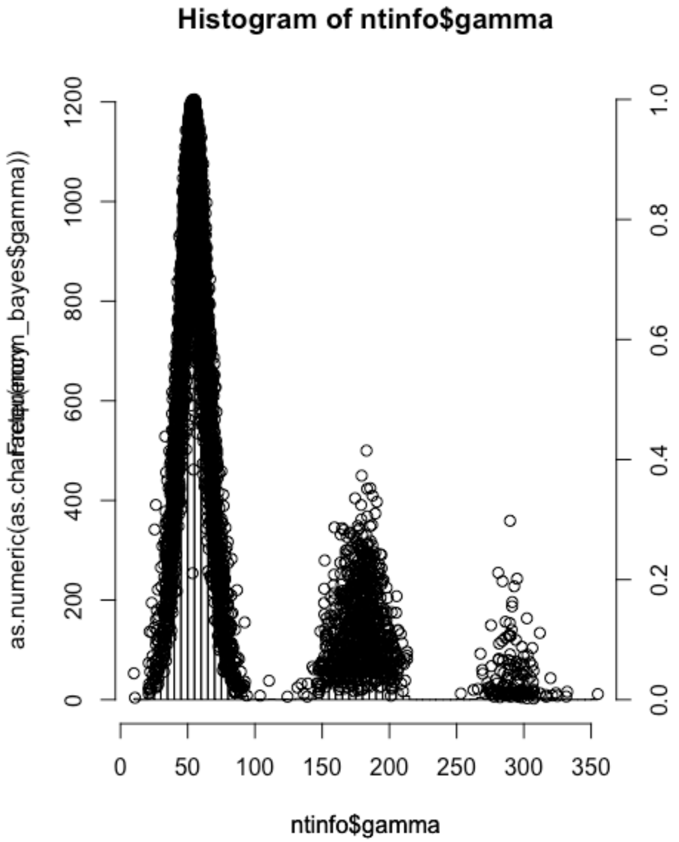
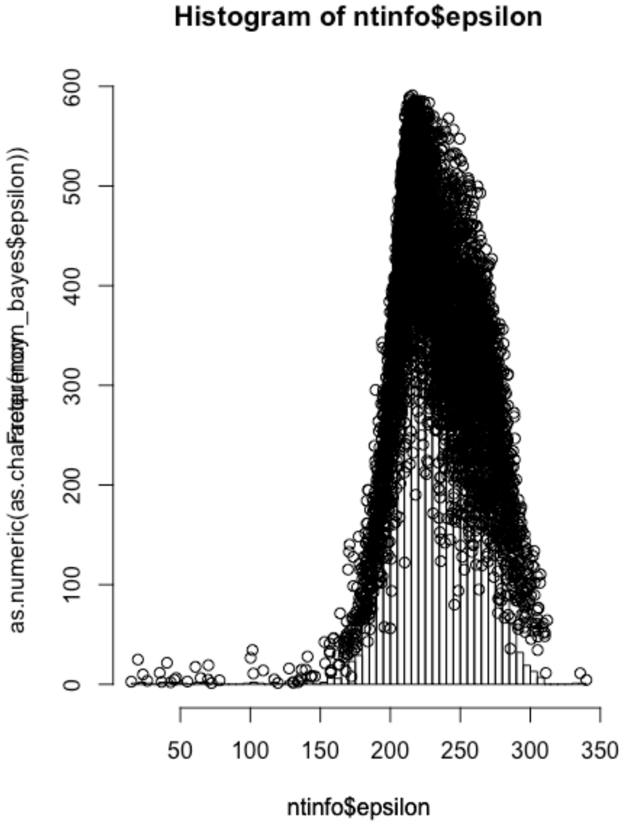
Check all the plots generated



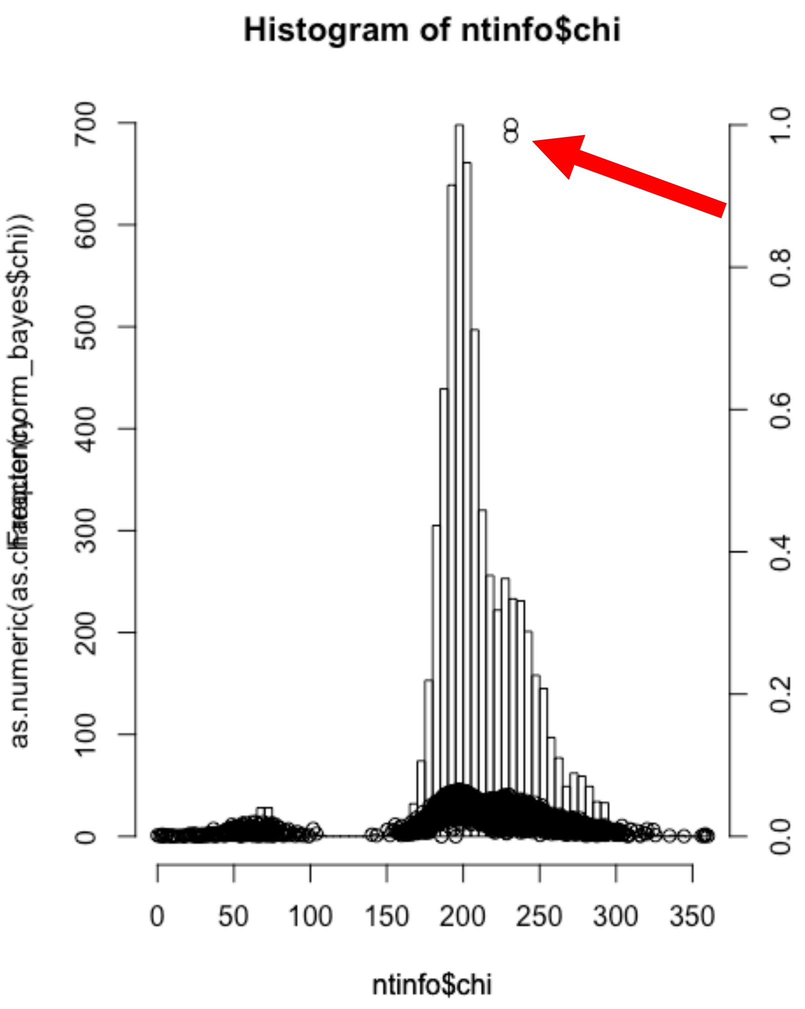
Check all the plots generated







Check all the plots generated



Stripe the highest values for beta and chi scores

·IDs for BETA:

771
5031

	alpha	beta	gamma	delta	epsilon	zeta	chi
771	289.201	186.393	64.264	144.276.	274.418	85.945	225.393
5031	305.376	186.394	57.141	159.383	292.826.	86.494	249.556

·Scoring by Bayes function:

	alpha	beta	gamma	delta	epsilon	zeta	chi	total
771	0.8973062	16.05909	0.5409483	0.4490677	0.47092283	0.3226892	0.6540827	2.770586
5031	0.5285648	15.77888	0.6109064	0.1482734	0.09670432	0.1870816	0.3746445	2.532151

·Scoring by cybeRNAting:

	alpha	beta	gamma	delta	epsilon	zeta	chi
1	0.922	0.262	0.441	0.015	0.024	0.012	0.079
2	0.375	0.335	0.836	0.02	0.021	0.017	0.051

·IDs for CHI:

1677
3233

	alpha	beta	gamma	delta	epsilon	zeta	chi
1677	270.223	201.464	67.534	135.958	246.685	304.819	232.028
3233	303.511	187.134	42.727	136.420	284.622	91.193	232.027

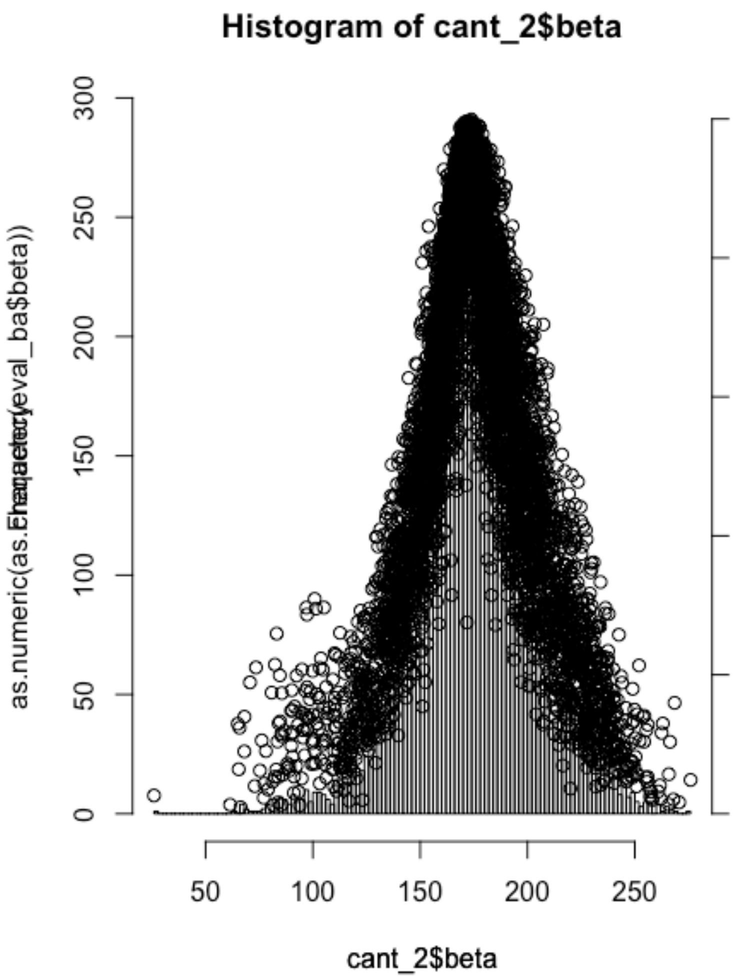
·Scoring by Bayes function:

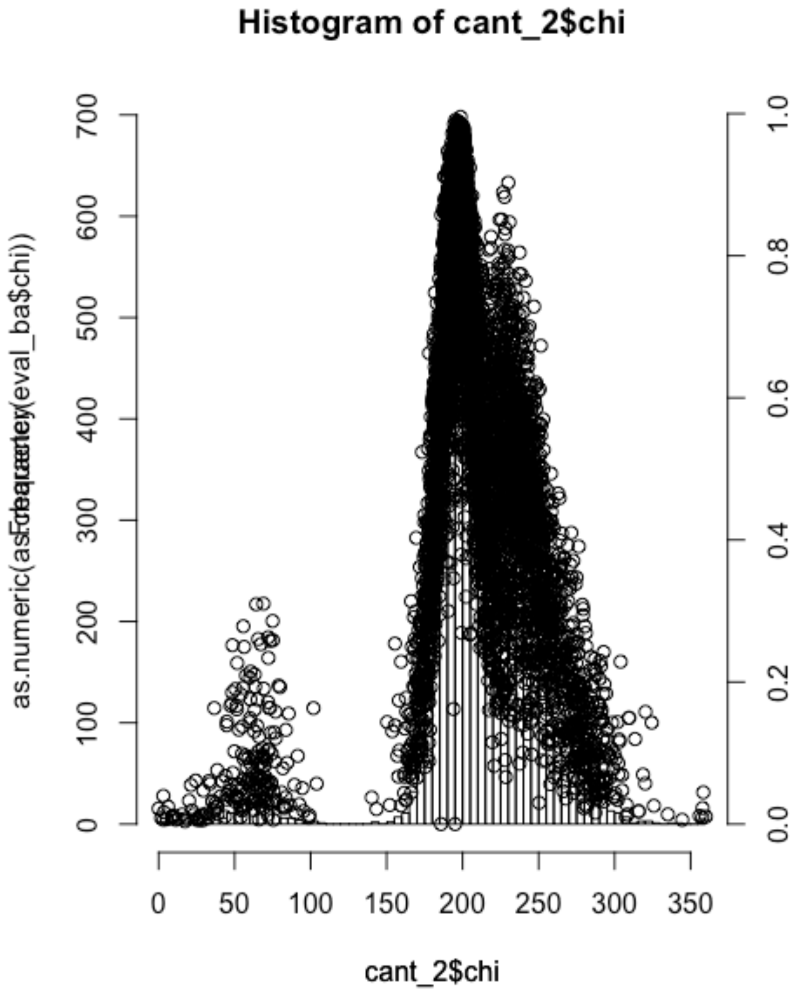
	alpha	beta	gamma	delta	epsilon	zeta	chi	total
1677	0.3167092	0.6932020	0.5414358	0.1373346	0.8894099	0.4993746	15.17293	2.607199
3233	0.5621434	0.6215669	0.5625761	0.1392141	0.3949913	0.3781720	15.41195	2.581517

·Scoring by cybeRNAting:

	alpha	beta	gamma	delta	epsilon	zeta	chi
1	0.138	0.087	0.29	0.047	0.109	0.128	0.094
2	0.477	0.311	0.205.	0.043	0.032.	0.018	0.073

Remove those 4 nucleotides and check again plots for Beta and Chi





We are already normalizing with the max_dens dividing so that our data fits in the range of [0,1]

	alpha	$\boxed{\text{beta}}$	gamma	delta	epsilon	zeta	chi	total
771	0.8973062	~~16.05909~~	0.5409483	0.4490677	0.47092283	0.3226892	0.6540827	2.770586
5031	0.5285648	~~15.[illegible]7888~~	0.6109064	0.1482734	0.09670432	0.1870816	0.3746445	2.532151

Does not make sense > 1 values

Deep analysis on the ID: 771. Solving its problem may solve problem for the others

	alpha	beta	gamma	delta	epsilon	zeta	chi
771	289.201	186.393	64.264	144.276.	274.418	85.945	225.393

1) Scoring by Bayes function:

	alpha	beta	gamma	delta	epsilon	zeta	chi	total
771	0.8973062	**16.05909**	0.5409483	0.4490677	0.47092283	0.3226892	0.6540827	2.770586

2) Analysis of the different contributions "beta"

```
beta_amb <- c(nulling(dnorms_calc("alpha",eval[,"alpha"],"beta",eval2[,"beta"])),
              nulling(dnorms_calc("gamma",eval[,"gamma"],"beta",eval2[,"beta"])),
              nulling(dnorms_calc("delta",eval[,"delta"],"beta",eval2[,"beta"])),
              nulling(dnorms_calc("epsilon",eval[,"epsilon"],"beta",eval2[,"beta"])),
              nulling(dnorms_calc("zeta",eval[,"zeta"],"beta",eval2[,"beta"])),
              nulling(dnorms_calc("chi",eval[,"chi"],"beta",eval2[,"beta"])))
```

OUTPUT: 0.7346722 0.9257825 0.7290086 0.8479728 $\boxed{92.1275127}$ 0.9895698

3) Check at the probability function for: zeta – beta pairs

```
#####For zeta
if(angle == "zeta"){
  zeta <- vector("list", length = length(inter_list))

  for(i in 1:length(inter_list)){
    index <- which(ntinfo_m[,"zeta"] == i)
    if(length(index) > 5){
      dataframe2 <- ntinfo_m[index, ]
      dataframe2_no_disc <- ntinfo[index,]

      #Check for zeta analysis
      gaussians_alpha<- fit_data(na.omit(dataframe2_no_disc$alpha))
      zeta[[i]]$alpha$prop <- gaussians_alpha$prop
      zeta[[i]]$alpha$mean <- gaussians_alpha$mean
      zeta[[i]]$alpha$sd <- gaussians_alpha$sd
      zeta[[i]]$alpha$max_dens <- gaussians_alpha$max_dens

      #Check for zeta
      gaussians_beta<- fit_data(na.omit(dataframe2_no_disc$beta))
      zeta[[i]]$beta$prop <- gaussians_beta$prop
      zeta[[i]]$beta$mean <- gaussians_beta$mean
      zeta[[i]]$beta$sd <- gaussians_beta$sd
      zeta[[i]]$beta$max_dens <- gaussians_beta$max_dens
```

4) Dataframe subset for: zeta = 86

```
        alpha    beta    gamma   delta epsilon   zeta     chi
150   299.985 224.918   58.309 152.590 275.147 86.087 186.036
771   289.201 186.393   64.264 144.276 274.418 85.945 225.393
1775  277.810 196.823   64.400 150.562 284.776 86.384 243.595
1969   63.340 182.985   69.275 144.326 270.723 86.313 237.007
1981  304.409 177.257   44.152  80.000 196.679 85.811 201.329
2632  270.536 178.374   69.283  85.215 278.259 85.647 200.619
3338   48.854 178.523  281.555 149.244 231.742 86.158  58.462
3446   73.771 202.162   59.702  93.546 245.278 85.703 293.130
3701   69.662 199.497   38.083 148.515 285.610 85.506  50.386
3864  297.686 179.831   50.869 148.359 242.312 86.123 241.039
4316   55.942 171.598  294.024 150.164 223.961 85.928  56.351
4490  291.336 175.557   63.874  88.631 264.995 85.927 204.186
4632  101.495 229.141   61.469 147.448 204.297 85.988 209.828
5031  305.376 186.394   57.141 159.383 292.826 86.494 249.556
5101  269.884 160.637   50.488 150.559 281.099 85.792 246.733
5994  289.182 201.943   51.546 148.065 273.845 85.614 238.402
6223   62.093 169.994   46.134 142.933 232.856 86.388 240.206
```

5) Fitted gaussians for beta when zeta = 86

```
        prop     mean         sd  max_dens
1 0.36354770 173.7021 7.27810908 0.6179944
2 0.05430858 176.5040 0.87766141 0.6179944
3 0.11202925 178.4486 0.07449993 0.6179944
4 0.11763805 186.3935 0.00050000 0.6179944
5 0.11744160 198.1713 1.34759935 0.6179944
6 0.11738776 202.0526 0.10949999 0.6179944
7 0.11764706 227.0295 2.11150000 0.6179944
```

6) Compute dnorm() calculus on each of them

4th Gaussian gives such bad result

```
> loopout(eval2 = 186.393, x = gaussians_beta, ind = 7)
[1] 0.007050696
[1] 7.050696e-03 1.080494e-29
[1] 7.050696e-03 1.080494e-29 0.000000e+00
[1] 7.050696e-03 1.080494e-29 0.000000e+00 9.212046e+01
[1] 7.050696e-03 1.080494e-29 0.000000e+00 9.212046e+01 1.452381e-18
[1] 7.050696e-03 1.080494e-29 0.000000e+00 9.212046e+01 1.452381e-18 0.000000e+00
[1] 7.050696e-03 1.080494e-29 0.000000e+00 9.212046e+01 1.452381e-18 0.000000e+00 1.343955e-82
[1] 92.12751
```

7) Compute dnorm() manually on the 4th row

```
> dnorm(x = 186.393, mean = 186.3935, sd = 0.00050000)
[1] 483.9414
> dnorm(x = 186.393, mean = 186.3935, sd = 0.00050000) * 0.11763805
[1] 56.92993
> dnorm(x = 186.393, mean = 186.3935, sd = 0.00050000) * 0.11763805 / 0.6179944
[1] 92.12046
```

PROBLEM:

·Standard deviation is too low so minimal changes on x versus mean create a lot of change at dnorm()

SOLUTION:

·Apply filtering for sd

·Data handling

******Trying with different values on x:**

```
> dnorm(x = 184, mean = 186.3935, sd = 0.00050000) * 0.11763805 / 0.6179944
[1] 0
> dnorm(x = 186, mean = 186.3935, sd = 0.00050000) * 0.11763805 / 0.6179944
[1] 0
> dnorm(x = 187, mean = 186.3935, sd = 0.00050000) * 0.11763805 / 0.6179944
[1] 0
> dnorm(x = 186.37, mean = 186.3935, sd = 0.00050000) * 0.11763805 / 0.6179944
[1] 0
> dnorm(x = 186.395, mean = 186.3935, sd = 0.00050000) * 0.11763805 / 0.6179944
[1] 1.687245
> dnorm(x = 186.3933, mean = 186.3935, sd = 0.00050000) * 0.11763805 / 0.6179944
[1] 140.2038
```