

# Bioconductor snpStats Bugs

Hin-Tak Leung

March 28, 2013

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Bugs in snpStats GLM estimates</b>	<b>3</b>
2.1	snpStats::snp.*hs.estimates() returns garbage . . . . .	3
<b>3</b>	<b>Multiple snpStats issues of data corruption, memory violations and crashes</b>	<b>5</b>
<b>4</b>	<b>Bugs in snpStats GLM score tests</b>	<b>5</b>
4.1	snpStats::snp.lhs.tests(...,robust=TRUE) returns garbage . . . . .	5
4.2	Malformed "GlmTests" S4 object from snpStats::snp.lhs.tests() . . . . .	7
4.3	Crazy large/negative number of samples from snpStats GLM tests . . . . .	8
4.4	snpStats::snp.rhs.tests() returning garbage with or without robust . . . . .	8
<b>5</b>	<b>2-df Bug in snpStats::single.snp.tests()</b>	<b>10</b>
<b>6</b>	<b>Inputs/Conversions of uncertain genotypes from posterior probabilities</b>	<b>13</b>
<b>7</b>	<b>2nd bug in Inputs/Conversions of uncertain genotypes from posterior probabilities</b>	<b>14</b>
<b>8</b>	<b>cbind/rbind in snpStats 1.7.4 onwards</b>	<b>14</b>
<b>9</b>	<b>ld() is broken, has always been broken</b>	<b>14</b>
<b>10</b>	<b>64-bit mode for snpStats::single.snp.tests()</b>	<b>14</b>
<b>11</b>	<b>X chromosome conversion</b>	<b>14</b>

## 1 Introduction

In the Regression and Migration vignette, we discovered that there has been a bug in snpMatrix's `single.snp.tests()` for many years, which can affect 1% to 2% of SNPs, and fixed it. This vignette uses the testsuite code in snpMatrix to reveal snpStats' bug(s). The quick summary is that most (all?) of the statistical tests were broken to various extent around October 2008 by the imputation-related changes. That is 3 years of flawed publications. David made an effort with

1.3.7+ (20th October 2011) but did not get very far. This document is usually built against current snpStats HEAD as well as 1.3.6 (17th October 2011), the latter because of the number of flawed results in 3 years.

Despite many routines being of the same names but behaving differently (such as the buggy `single.snp.tests()` in `snpStats` vs the correct one in `snpMatrix`) and a warning about routines shadowing each other, it is possible to use either in the same R session even in alternating statements, as long as either are referenced explicitly in each step. This usually consists of prefix'ing with explicit namespace references (e.g. `snpMatrix::single.snp.tests()` instead of `single.snp.tests()`) or adding `package=` within.

```
> library(snpMatrix)
> library(snpStats)
> sessionInfo()
```

```
R version 2.15.3 (2013-03-01)
Platform: i686-redhat-linux-gnu (32-bit)
```

```
locale:
```

```
[1] LC_CTYPE=en_GB.utf8      LC_NUMERIC=C
[3] LC_TIME=en_GB.utf8       LC_COLLATE=en_GB.utf8
[5] LC_MONETARY=en_GB.utf8   LC_MESSAGES=en_GB.utf8
[7] LC_PAPER=C               LC_NAME=C
[9] LC_ADDRESS=C            LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_GB.utf8 LC_IDENTIFICATION=C
```

```
attached base packages:
```

```
[1] grDevices datasets splines graphics utils stats methods
[8] base
```

```
other attached packages:
```

```
[1] snpStats_1.9.3          snpMatrix_1.19.0.18 Matrix_1.0-11
[4] lattice_0.20-13        survival_2.37-2
```

```
loaded via a namespace (and not attached):
```

```
[1] BiocGenerics_0.4.0 grid_2.15.3          tools_2.15.3
```

At this point there is a warning:

```
Attaching package: 'snpStats'
```

```
The following object(s) are masked from 'package:snpMatrix':
```

```
can.impute, chi.squared, col.summary, deg.freedom, effect.sign,
effective.sample.size, filter.rules, Fst, glm.test.control,
ibsCount, ibsDist, imputation.maf, imputation.nsnp, imputation.r2,
impute.snps, ld, misinherits, mvtests, p.value, plotUncertainty,
```

```
pool, pool2, pp, qq.chisq, read.beagle, read.impute, read.mach,
read.pedfile, read.plink, read.snps.long, row.summary, sample.size,
single.snp.tests, snp.cbind, snp.cor, snp.imputation,
snp.lhs.estimate, snp.lhs.tests, snp.post.multiply,
snp.pre.multiply, snp.rbind, snp.rhs.estimate, snp.rhs.tests,
switch.alleles, tdt.snp, test.allele.switch, write.plink, xxt
```

Loading `snpMatrix124` simultaneously can also cause a few warnings of the form, understandably:

```
A specification for class X.snp.matrix in package 'snpMatrix124'
seems equivalent to one from package 'snpMatrix' and is not turning
on duplicate class definitions for this class
```

The warning is harmless, if one pays attention to specifying each explicitly, as below.

## 2 Bugs in `snpStats` GLM estimates

### 2.1 `snpStats::snp.*hs.estimate()` returns garbage

`snp.*hs.estimate()` gives garbage — the way to illustrate this is simply running the corresponding `snp.rhs.tests` and compare:

```
> data(testdata, package = "snpStats")
> test2 <- snpStats::snp.rhs.estimate(cc ~ region + sex, family = "binomial",
+   data = subject.data, snp.data = Autosomes, sets = 1:10)
> test2.t <- snpStats::snp.rhs.tests(cc ~ region + sex, family = "binomial",
+   data = subject.data, snp.data = Autosomes, tests = 1:10)
> print(cbind(as(as(test2, "GlmTests"), "data.frame"), as(test2.t,
+   "data.frame")))
```

	Chi.squared	Df	p.value	Chi.squared	Df	p.value
173760	0.002022755	1	0.96412719	0.96172092	1	0.32675367
173761	1.614689875	1	0.20383380	1.61954459	1	0.20315530
173762	2.052781924	1	0.15192836	2.05991420	1	0.15121869
173767	0.777421401	1	0.37793093	0.77858708	1	0.37757361
173769	2.762354385	1	0.09650612	2.92552940	1	0.08718862
173770	NA	NA	NA	NA	0	NA
173772	0.002046886	1	0.96391400	1.02511037	1	0.31130988
173774	0.729732211	1	0.39297000	0.73179195	1	0.39230296
173775	0.952718263	1	0.32902835	0.95584241	1	0.32823661
173776	0.090165195	1	0.76396725	0.09019184	1	0.76393342

This is the correct result from `snpMatrix` (1.17.7.11) (Wald test close to the score test):

```

> Autosomes <- new("snp.matrix", Autosomes@.Data)
> test2 <- snpMatrix::snp.rhs.estimates(cc ~ region + sex, family = "binomial",
+   data = subject.data, snp.data = Autosomes, sets = 1:10)
> test2.t <- snpMatrix::snp.rhs.tests(cc ~ region + sex, family = "binomial",
+   data = subject.data, snp.data = Autosomes, tests = 1:10)
> print(cbind(as(as(test2, "snp.tests.glm"), "data.frame"), as(test2.t,
+   "data.frame"))))

```

	Chi.squared	Df	p.value	Chi.squared	Df	p.value
173760	0.68794335	1	0.40686481	0.96171330	1	0.32675559
173761	1.61514861	1	0.20376957	1.61953853	1	0.20315614
173762	2.05337857	1	0.15186885	2.05990584	1	0.15121951
173767	0.77747875	1	0.37791334	0.77858708	1	0.37757361
173769	2.76587572	1	0.09629398	2.92549240	1	0.08719062
173770	NA	NA	NA	NA	0	NA
173772	0.71127227	1	0.39902177	1.02511036	1	0.31130988
173774	0.72973283	1	0.39296980	0.73179195	1	0.39230296
173775	0.95271902	1	0.32902816	0.95584109	1	0.32823694
173776	0.09016525	1	0.76396718	0.09019184	1	0.76393341

See another garbage result from snpStats:

```

> data(testdata, package = "snpStats")
> test3 <- snpStats::snp.lhs.estimates(Autosomes[, 1:10], ~strata(cc),
+   ~strata(region), data = subject.data, robust = TRUE)
> test3.t <- snpStats::snp.lhs.tests(Autosomes[, 1:10], ~strata(cc),
+   ~strata(region), data = subject.data, robust = TRUE)
> print(cbind(as(as(test3, "GlmTests"), "data.frame"), as(test3.t,
+   "data.frame"))))

```

	Chi.squared	Df	p.value	Chi.squared	Df	p.value
173760	NA	NA	NA	NaN	0	NaN
173761	NA	NA	NA	10.263701	9	0.32956204
173762	NA	NA	NA	10.476287	9	0.31331934
173767	NA	NA	NA	15.045105	9	0.08970445
173769	NA	NA	NA	16.267175	9	0.06150760
173770	NA	NA	NA	0.000000	0	1.00000000
173772	NA	NA	NA	NaN	0	NaN
173774	NA	NA	NA	9.411857	9	0.40015665
173775	NA	NA	NA	13.440413	9	0.14366942
173776	NA	NA	NA	16.797680	9	0.05198017

At the time of this writing, snpMatrix (1.17.7.11, unreleased) isn't correct either, but better:

```

> Autosomes <- new("snp.matrix", Autosomes@.Data)
> test3 <- snpMatrix::snp.lhs.estimates(Autosomes[, 1:10], ~strata(cc),
+   ~strata(region), data = subject.data, robust = TRUE)

```

```

> test3.t <- snpMatrix:::snp.lhs.tests(Autosomes[, 1:10], ~strata(cc),
+   ~strata(region), data = subject.data, robust = TRUE)
> print(cbind(as(as(test3, "snp.tests.glm"), "data.frame"), as(test3.t,
+   "data.frame")))

```

	Chi.squared	Df	p.value	Chi.squared	Df	p.value
173760	1004.549087	9	1.801433e-210	1.009134	9	0.99941616
173761	10.959622	9	2.784868e-01	10.263701	9	0.32956204
173762	11.155682	9	2.651885e-01	10.476287	9	0.31331934
173767	17.009456	9	4.856810e-02	15.045105	9	0.08970445
173769	201.767706	9	1.411265e-38	16.267175	9	0.06150760
173770	NA	NA	NA	0.000000	0	1.00000000
173772	5.449466	9	7.935005e-01	1.009439	9	0.99941544
173774	278.130321	9	1.125582e-54	9.411857	9	0.40015665
173775	337.152659	9	3.352980e-67	13.440413	9	0.14366942
173776	426.044143	9	3.773047e-86	16.797680	9	0.05198017

### 3 Multiple snpStats issues of data corruption, memory violations and crashes

There are multiple issues of data corruption, memory violation and crashes in the GLM related code. The best way to demonstrate this is turn on `gctorture()` and uses the GLM score tests/estimates and see R crash.

## 4 Bugs in snpStats GLM score tests

### 4.1 snpStats::snp.lhs.tests(...,robust=TRUE) returns garbage

```

> data(testdata, package = "snpStats")
> snpStats:::snp.lhs.tests(Autosomes[, 1:10], ~strata(cc), ~strata(region),
+   data = subject.data, robust = TRUE)

```

	Chi.squared	Df	p.value
173760	NaN	0	NaN
173761	10.263701	9	0.32956204
173762	10.476287	9	0.31331934
173767	15.045105	9	0.08970445
173769	16.267175	9	0.06150760
173770	0.000000	0	1.00000000
173772	NaN	0	NaN
173774	9.411857	9	0.40015665
173775	13.440413	9	0.14366942
173776	16.797680	9	0.05198017

The correct result should be somewhat close to the non-robust result:

```
> snpStats::snp.lhs.tests(Autosomes[, 1:10], ~strata(cc), ~strata(region),
+   data = subject.data, robust = FALSE)
```

	Chi.squared	Df	p.value
173760	NaN	9	NaN
173761	12.272003	9	0.1984058
173762	12.476151	9	0.1877771
173767	14.462676	9	0.1067926
173769	8.589652	9	0.4759812
173770	0.000000	0	1.0000000
173772	NaN	9	NaN
173774	6.156140	9	0.7241952
173775	8.812111	9	0.4547958
173776	7.602749	9	0.5746207

Now we convert snpStats classes (mixed-cases without ".") to snpMatrix's (lowercases with "."), and re-run the the snpMatrix version of `snp.lhs.tests()`:

```
> snpMatrix::snp.lhs.tests(new("snp.matrix", Autosomes@.Data)[,
+   1:10], ~strata(cc), ~strata(region), data = subject.data,
+   robust = TRUE)
```

	Chi.squared	Df	p.value
173760	1.009134	9	0.99941616
173761	10.263701	9	0.32956204
173762	10.476287	9	0.31331934
173767	15.045105	9	0.08970445
173769	16.267175	9	0.06150760
173770	0.000000	0	1.00000000
173772	1.009439	9	0.99941544
173774	9.411857	9	0.40015665
173775	13.440413	9	0.14366942
173776	16.797680	9	0.05198017

Just to see what snpMatrix does without robust:

```
> snpMatrix::snp.lhs.tests(new("snp.matrix", Autosomes@.Data)[,
+   1:10], ~strata(cc), ~strata(region), data = subject.data,
+   robust = FALSE)
```

	Chi.squared	Df	p.value
173760	5.042974	9	0.8305461
173761	12.272003	9	0.1984058
173762	12.476151	9	0.1877771
173767	14.462676	9	0.1067926
173769	8.589652	9	0.4759812
173770	0.000000	0	1.0000000

```

173772    7.714940  9 0.5631090
173774    6.156140  9 0.7241952
173775    8.812111  9 0.4547958
173776    7.602749  9 0.5746207

```

This bug also exist in snpMatrix prior to 1.17.5.10 and was introduced with the imputation-related changes around October 2008.

See also how snpMatrix124 does:

```

> snpMatrix124::snp.lhs.tests(new("snp.matrix", Autosomes@.Data)[,
+   1:10], ~strata(cc), ~strata(region), data = subject.data,
+   robust = TRUE)

```

	Chi.squared	Df	Df.residual
173760	1.008943	8	198
173761	10.263701	9	398
173762	10.476287	9	396
173767	15.045105	9	376
173769	16.006621	8	394
173770	NA	NA	NA
173772	1.009439	8	199
173774	9.411857	9	386
173775	13.440413	9	397
173776	16.797680	9	398

```

> snpMatrix124::snp.lhs.tests(new("snp.matrix", Autosomes@.Data)[,
+   1:10], ~strata(cc), ~strata(region), data = subject.data,
+   robust = FALSE)

```

	Chi.squared	Df	Df.residual
173760	5.042974	9	198
173761	12.272003	9	398
173762	12.476151	9	396
173767	14.462676	9	376
173769	8.589652	9	394
173770	NA	NA	NA
173772	7.714940	9	199
173774	6.156140	9	386
173775	8.812111	9	397
173776	7.602749	9	398

## 4.2 Malformed "GlmTests" S4 object from snpStats::snp.lhs.tests()

Upto and including snpStats 1.3.6:

```

> result <- snpStats::snp.lhs.tests(Autosomes[, 1:10], ~strata(cc),
+   ~strata(region), data = subject.data)

```

Looking at `result` gives a hard error so I'll just show the message below:

```
> str(result)
Error in FUN(c("snp.names", "var.names", "chisq", "df", "N")[[2L]], ...) :
  no slot of name "var.names" for this object of class "GlmTests"
```

There is no need to show the alternatives as this is clearly broken.  
This bug is specific to `snpStats` and has no equivalent in `snpMatrix`.

### 4.3 Crazy large/negative number of samples from `snpStats` GLM tests

Upto and including `snpStats` 1.3.6:

```
> result@N

[1] 202248192 202248160 202248128 202248096 202248064 202248032 202248000
[8] 202247968 202247936 202247904
```

Hundred million samples and negative number of samples?

This bug also exist in `snpMatrix` prior to 1.17.5.10 and was introduced with the imputation-related changes around October 2008.

### 4.4 `snpStats::snp.rhs.tests()` returning garbage with or without robust

```
> snpStats::snp.rhs.tests(cc ~ strata(region, sex), family = "binomial",
+   data = subject.data, snp.data = Autosomes, tests = 1:10)
```

	Chi.squared	Df	p.value
173760	1.25356125	1	0.26287339
173761	1.61290542	1	0.20408387
173762	2.04226350	1	0.15298186
173767	NaN	1	NaN
173769	3.54351327	1	0.05977872
173770	NA	0	NA
173772	0.59863946	1	0.43909761
173774	0.82443150	1	0.36388768
173775	0.87744532	1	0.34890234
173776	0.09218633	1	0.76141588

The correct result shouldn't be too far from without sex:

```
> snpStats::snp.rhs.tests(cc ~ strata(region), family = "binomial",
+   data = subject.data, snp.data = Autosomes, tests = 1:10)
```



	Chi.squared	Df	p.value
173760	1.01538462	1	0.31361630
173761	1.46259571	1	0.22651757
173762	1.92028786	1	0.16582493
173767	0.77609738	1	0.37833736
173769	2.92614948	1	0.08715513
173770	NA	0	NA
173772	1.11008326	1	0.29206385
173774	0.66697270	1	0.41410906
173775	0.96730037	1	0.32535438
173776	0.09831885	1	0.75385649

Here is how snpMatrix does it:

```
> snpMatrix::snp.rhs.tests(cc ~ strata(region, sex), family = "binomial",
+   data = subject.data, snp.data = new("snp.matrix", Autosomes@.Data),
+   tests = 1:10)
```

	Chi.squared	Df	p.value
173760	1.25356125	1	0.26287339
173761	1.61290542	1	0.20408387
173762	2.04226350	1	0.15298186
173767	0.29671726	1	0.58594776
173769	3.54351327	1	0.05977872
173770	NA	0	NA
173772	0.59863946	1	0.43909761
173774	0.82443150	1	0.36388768
173775	0.87744532	1	0.34890234
173776	0.09218633	1	0.76141588

Just to see that snpMatrix does it without sex:

```
> snpMatrix::snp.rhs.tests(cc ~ strata(region), family = "binomial",
+   data = subject.data, snp.data = new("snp.matrix", Autosomes@.Data),
+   tests = 1:10)
```

	Chi.squared	Df	p.value
173760	1.01538462	1	0.31361630
173761	1.46259571	1	0.22651757
173762	1.92028786	1	0.16582493
173767	0.77609738	1	0.37833736
173769	2.92614948	1	0.08715513
173770	NA	0	NA
173772	1.11008326	1	0.29206385
173774	0.66697270	1	0.41410906
173775	0.96730037	1	0.32535438
173776	0.09831885	1	0.75385649

This bug also exist in snpMatrix prior to 1.17.5.10 and was introduced with the imputation-related changes around October 2008.

See how snpMatrix124 does:

```
> snpMatrix124::snp.rhs.tests(cc ~ strata(region, sex), family = "binomial",
+   data = subject.data, snp.data = new("snp.matrix", Autosomes@.Data),
+   tests = 1:10)
```

	Chi.squared	Df	Df.residual
173760	1.25356125	1	373
173761	1.61290542	1	376
173762	2.04226350	1	374
173767	0.29671726	1	350
173769	3.54958407	1	372
173770	NA	NA	376
173772	0.59863946	1	376
173774	0.82443150	1	365
173775	0.87744532	1	375
173776	0.09218633	1	376

```
> snpMatrix124::snp.rhs.tests(cc ~ strata(region), family = "binomial",
+   data = subject.data, snp.data = new("snp.matrix", Autosomes@.Data),
+   tests = 1:10)
```

	Chi.squared	Df	Df.residual
173760	1.01538462	1	387
173761	1.46259571	1	390
173762	1.92028786	1	388
173767	0.77609738	1	368
173769	2.92614948	1	386
173770	NA	NA	390
173772	1.11008326	1	390
173774	0.66697270	1	378
173775	0.96730037	1	389
173776	0.09831885	1	390

## 5 2-df Bug in snpStats::single.snp.tests()

```
> data(for.exercise, package = "snpStats")
> tests.snpStats <- snpStats::single.snp.tests(cc, stratum, data = subject.support,
+   snp.data = snps.10)
```

Now we convert snpStats classes (mixed-cases without “.”) to snpMatrix’s (lowercases with “.”), and re-run the the snpMatrix version of single.snp.tests():

```
> str(snps.10)
```

```
Formal class 'SnpMatrix' [package "snpStats"] with 1 slots
..@ .Data: raw [1:1000, 1:28501] 01 01 01 01 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:1000] "jpt.869" "jpt.862" "jpt.948" "ceu.564" ...
.. .. ..$ : chr [1:28501] "rs7909677" "rs7093061" "rs12773042" "rs7475011" ...
```

```
> snps.10 <- new("snp.matrix", snps.10@.Data)
> str(snps.10)
```

```
Formal class 'snp.matrix' [package "snpMatrix"] with 1 slots
..@ .Data: raw [1:1000, 1:28501] 01 01 01 01 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:1000] "jpt.869" "jpt.862" "jpt.948" "ceu.564" ...
.. .. ..$ : chr [1:28501] "rs7909677" "rs7093061" "rs12773042" "rs7475011" ...
```

```
> tests.snpMatrix <- snpMatrix::single.snp.tests(cc, stratum, data = subject.support,
+       snp.data = snps.10)
```

Then we use the testsuite code to compare:

```
> all.equal(tests.snpStats@chisq[, "1 df"], tests.snpMatrix@chisq[,
+       "1 df"], tolerance = 0)
```

```
[1] TRUE
```

```
> all.equal(tests.snpStats@chisq[, "2 df"], tests.snpMatrix@chisq[,
+       "2 df"], tolerance = 0)
```

```
[1] "'is.NA' value mismatch: 789 in current 807 in target"
```

```
> snpMatrix:::chi2.all.equal(tests.snpStats@chisq[, "2 df"], tests.snpMatrix@chisq[,
+       "2 df"])
```

```
Max absolute finite difference: 0
Max relative finite difference: 0
Finite in 1st but not in 2nd: 0
Finite in 2nd but not in 1st: 18
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.8604  1.2830  2.4020  2.3000  2.9840  4.2580
Difference
      18
```

Or 20 SNP tests failed in snpStats but okay in snpMatrix.

Now we run the non-stratified tests, but convert in the opposite direction, and compare:

```
> tests.snpMatrix.crude <- snpMatrix::single.snp.tests(cc, data = subject.support,
+       snp.data = snps.10)
> str(snps.10)
```

```
Formal class 'snp.matrix' [package "snpMatrix"] with 1 slots
..@ .Data: raw [1:1000, 1:28501] 01 01 01 01 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:1000] "jpt.869" "jpt.862" "jpt.948" "ceu.564" ...
.. .. ..$ : chr [1:28501] "rs7909677" "rs7093061" "rs12773042" "rs7475011" ...
```

```
> snps.10 <- new("SnpMatrix", snps.10@.Data)
> str(snps.10)
```

```
Formal class 'SnpMatrix' [package "snpStats"] with 1 slots
..@ .Data: raw [1:1000, 1:28501] 01 01 01 01 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:1000] "jpt.869" "jpt.862" "jpt.948" "ceu.564" ...
.. .. ..$ : chr [1:28501] "rs7909677" "rs7093061" "rs12773042" "rs7475011" ...
```

```
> tests.snpStats.crude <- snpStats::single.snp.tests(cc, data = subject.support,
+   snp.data = snps.10)
> all.equal(tests.snpStats.crude@chisq[, "2 df"], tests.snpMatrix.crude@chisq[,
+   "2 df"], tolerance = 0)
```

```
[1] "'is.NA' value mismatch: 789 in current 807 in target"
```

```
> snpMatrix:::chi2.all.equal(tests.snpStats.crude@chisq[, "2 df"],
+   tests.snpMatrix.crude@chisq[, "2 df"])
```

```
Max absolute finite difference: 0
Max relative finite difference: 0
Finite in 1st but not in 2nd: 0
Finite in 2nd but not in 1st: 18
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.043  1.408   2.405   2.323   2.865   3.922
Difference
      18
```

Or 18 SNP tests failed in snpStats but okay in snpMatrix.

This bug also exist in snpMatrix prior to 1.17.4.9 for its entire history (i.e. since pre-1.0), and differently before 1.5.x also.

This explains why snpStats and snpMatrix124 differs:

```
> snps.10 <- new("snp.matrix", snps.10@.Data)
> tests.snpMatrix124 <- snpMatrix124::single.snp.tests(cc, stratum,
+   data = subject.support, snp.data = snps.10)
> all.equal(tests.snpStats@chisq[, "1 df"], tests.snpMatrix124$chi2.1df,
+   tolerance = 0)
```

```
[1] "Mean relative difference: 1.790103e-16"
```

```

> all.equal(tests.snpStats@chisq[, "2 df"], tests.snpMatrix124$chi2.2df,
+   tolerance = 0)

[1] "'is.NA' value mismatch: 789 in current 807 in target"

> snpMatrix:::chi2.all.equal(tests.snpMatrix124$chi2.1df, tests.snpMatrix@chisq[,
+   "1 df"])

Max absolute finite difference: 7.105427e-15
Max relative finite difference: 8.941748e-13
Finite in 1st but not in 2nd: 0
Finite in 2nd but not in 1st: 0
Difference
      0

> snpMatrix:::chi2.all.equal(tests.snpStats@chisq[, "2 df"], tests.snpMatrix124$chi2.2df)

Max absolute finite difference: 3.819167e-14
Max relative finite difference: 2.35139e-14
Finite in 1st but not in 2nd: 0
Finite in 2nd but not in 1st: 18
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.8604  1.2830  2.4020  2.3000  2.9840  4.2580
Difference
      18

> snpMatrix:::chi2.all.equal(tests.snpMatrix124$chi2.2df, tests.snpMatrix@chisq[,
+   "2 df"])

Max absolute finite difference: 3.819167e-14
Max relative finite difference: 4.193517e-14
Finite in 1st but not in 2nd: 0
Finite in 2nd but not in 1st: 0
Difference
      0

```

So out of 20 of those SNPs that snpStats failed, snpMatrix124 can do 18 and much closer to snpMatrix 1.17.6.10+ .

## 6 Inputs/Conversions of uncertain genotypes from posterior probabilities

There is a compiler/optimization-dependent bug in inputs/conversions of uncertain genotypes from posterior probabilities, where basically garbage-in, garbage-out. It is seen on Linux/gcc, and possibly all gcc-based systems at normal usage. This means all common systems, since R on windows also uses gcc.

It is fixed in snpMatrix 1.19.0.14 onwards (although snpMatrix does not use uncertain genotypes anywhere up to and including 1.19.0.14), and in snpStats x.x.x.2, 1.7.3.2 being the lowest version with the fix.

## 7 2nd bug in Inputs/Conversions of uncertain genotypes from posterior probabilities

Another bug related to inputs/conversions of uncertain genotypes from posterior probabilities was fixed with `snpStats` x.x.x.5 (1.9.3.5), from the development leading up to `snpMatrix` 1.19.0.18 . About 9.3% of wide-spectrum data are mis-read in `snpStats` x.x.x; in read-life data (Chiamo's bundled examples, 170,000 data points from WTCCC1) about 1% is mis-read.

`snpMatrix` 1.19.0.18 (April 2013) adds the `uncertain` option to `read.snps.chiamo()` for generating uncertain genotypes.

## 8 cbind/rbind in snpStats 1.7.4 onwards

`cbind/rbind` in `snpStats` 1.7.4 was rewritten. It does not work under some of the normal use-cases.

Incidentally, it was me who wrote the relevant part of R's `cbind/rbind`<sup>1</sup> in January 2006, as well as sub-assignments<sup>2</sup> of RAW types.

## 9 ld() is broken, has always been broken

The `ld()` function introduced in `snpStats` 1.1.8 (2011-02-15) is broken, and has always been broken. This was first discovered with `snpStats` 1.9.0<->1.9.1 which changes it slightly. Before or after the change, some snps with non-zero call rates could have NA for  $r^2$ . ( $r^2$  at worst is zero for uncorrelated but called snps). Wrong answers affect up to 5% of SNP pairs in typical datasets (e.g. hapmap).

The older `ld.snps()` function in `snpMatrix` is not affected. It was tested against haploview in 2006. As a result, haploview gained an enhancement in this area — for easier comparison! (Sun Apr 23 13:19:26 2006, “cache and show last popup” — the forwarded patch was committed by Jeff), and I gained commit rights to HaploView's repository and tagged HTL\_POST\_SERIALVER\_CHANGE, HTL\_PRE\_SERIALVER\_CHANGE, committed a major clean-up which touched almost half of the files (49 out of 102, to be precise) (Mon Feb 11 19:21:34 2008, “Talked to Jeff Barrett a while ago about this...”, committed myself) of HaploView. See Haploview's CVS commit logs for details.

## 10 64-bit mode for snpStats::single.snp.tests()

This returns all zeros in 64-bit machine occasionally (corruption?):

```
snp.lhs.tests(Autosomes[,1:10], ~cc, ~region, data=subject.data)
```

## 11 X chromosome conversion

The corresponding X-chromosome conversion is as follows:

<sup>1</sup>Bug 8529 - `rbind/cbind` unimplemented for raw (RAWSXP) types. [https://bugs.r-project.org/bugzilla3/show\\_bug.cgi?id=8529](https://bugs.r-project.org/bugzilla3/show_bug.cgi?id=8529)

<sup>2</sup>Bug 8530 - `sub*` assignment unimplemented for raw (RAWSXP) types. [https://bugs.r-project.org/bugzilla3/show\\_bug.cgi?id=8530](https://bugs.r-project.org/bugzilla3/show_bug.cgi?id=8530)

```

> data(testdata, package = "snpStats")
> str(Xchromosome)

Formal class 'XSnpmatrix' [package "snpStats"] with 2 slots
..@ .Data : raw [1:400, 1:155] 03 03 03 01 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:400] "1987" "436" "762" "1199" ...
.. .. ..$ : chr [1:155] "174193" "174196" "174197" "174208" ...
..@ diploid: Named logi [1:400] TRUE FALSE TRUE FALSE FALSE FALSE ...
.. ..- attr(*, "names")= chr [1:400] "1987" "436" "762" "1199" ...

> Xchromosome <- new("X.snp.matrix", Xchromosome@.Data, Female = Xchromosome@diploid)
> str(Xchromosome)

Formal class 'X.snp.matrix' [package "snpMatrix"] with 2 slots
..@ .Data : raw [1:400, 1:155] 03 03 03 01 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:400] "1987" "436" "762" "1199" ...
.. .. ..$ : chr [1:155] "174193" "174196" "174197" "174208" ...
..@ Female: Named logi [1:400] TRUE FALSE TRUE FALSE FALSE FALSE ...
.. ..- attr(*, "names")= chr [1:400] "1987" "436" "762" "1199" ...

```