```
homozygote(c104t, "C")TRUE    4.46  2.3e-05 ***
allele.count(a1691g, "G")    -0.77     0.44
c2249tT/C                     0.62     0.53
c2249tT/T                     0.59     0.55
---
Signif. codes:  0 '***' 0.001 '**' 0.01
                '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.1 on 95 degrees of
                         freedom
Multiple R-Squared: 0.176,
Adjusted R-squared: 0.141
F-statistic: 5.06 on 4 and 95 DF,
             p-value: 0.000969
```

Conclusion

The current release of the **genetics** package, 1.0.0, provides a complete set of classes and methods for handling single-locus genetic data as well as functions for computing and testing for departure from Hardy-Weinberg and linkage disequilibrium using a variety of estimators.

As noted earlier, Friedrich Leisch and I collaborated on the design of the data structures. While I was primarily motivated by the desire to provide a natural way to include single-locus genetic variables in statistical models, Fritz also wanted to support multiple genetic changes spread across one or more genes. As of the current version, my goal has largely been realized, but more work is necessary to fully support Fritz's goal.

In the future I intend to add functions to perform haplotype imputation and generate standard genetics plots.

I would like to thank Friedrich Leisch for his assistance in designing the genotype data structure, David Duffy for contributing the code for the `gregarious` and HWE.exact functions, and Michael Man for error reports and helpful discussion.

I welcome comments and contributions.

Gregory R. Warnes
Pfizer Global Research and Development
gregory_r_warnes@groton.pfizer.com

Variance Inflation Factors

by Jürgen Groß

A short discussion of centered and uncentered variance inflation factors. It is recommended to report both types of variance inflation for a linear model.

Centered variance inflation factors

A popular method in the classical linear regression model

$$\underset{n\times p}{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},\ \boldsymbol{X} = (\mathbf{1}_n, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_p),\ \boldsymbol{\varepsilon} \sim (\mathbf{0}, \sigma^2 \boldsymbol{I}_n),$$

to diagnose collinearity is to measure how much the variance $\mathrm{var}(\widehat{\boldsymbol{\beta}}_j)$ of the j-th element $\widehat{\boldsymbol{\beta}}_j$ of the ordinary least squares estimator $\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}$ is inflated by near linear dependencies of the columns of $\boldsymbol{X}$. This is done by putting $\mathrm{var}(\widehat{\boldsymbol{\beta}}_j)$ in relation to the variance of $\widehat{\boldsymbol{\beta}}_j$ as it would have been when

(a) the nonconstant columns $\boldsymbol{x}_2, \ldots, \boldsymbol{x}_p$ of $\boldsymbol{X}$ had been centered and

(b) the centered columns had been orthogonal to each other.

This variance is given by

$$v_j = \frac{\sigma^2}{\boldsymbol{x}_j'\boldsymbol{C}\boldsymbol{x}_j}, \quad \boldsymbol{C} = \boldsymbol{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n',$$

where $\boldsymbol{C}$ is the usual centering matrix. Then the *centered variance inflation factor* for the j-th variable is

$$\mathrm{VIF}_j = \frac{\mathrm{var}(\widehat{\boldsymbol{\beta}}_j)}{v_j} = \sigma^{-2}\,\mathrm{var}(\widehat{\boldsymbol{\beta}}_j)\boldsymbol{x}_j'\boldsymbol{C}\boldsymbol{x}_j, \quad j = 2, \ldots, p.$$

It can also be written in the form

$$\mathrm{VIF}_j = \frac{1}{1 - \widetilde{R}_j^2},$$

where

$$\widetilde{R}_j^2 = 1 - \frac{\boldsymbol{x}_j'\boldsymbol{M}_j\boldsymbol{x}_j}{\boldsymbol{x}_j'\boldsymbol{C}\boldsymbol{x}_j}, \quad \boldsymbol{M}_j = \boldsymbol{I}_n - \boldsymbol{X}_j(\boldsymbol{X}_j'\boldsymbol{X}_j)^{-1}\boldsymbol{X}_j',$$

is the centered coefficient of determination when the variable x_j is regressed on the remaining independent variables (including the intercept). The matrix $\boldsymbol{X}_j$ is obtained from $\boldsymbol{X}$ by deleting its j-th column. From this it is also intuitively clear that the centered VIF only works satisfactory in a model *with* intercept.

The centered variance inflation factor (as well as a generalized variance-inflation factor) is implemented in the R package **car** by Fox (2002), see also Fox (1997), and can also be computed with the function listed below, which has been communicated by Venables (2002) via the r-help mailing list.

```
> vif <- function(object, ...)
    UseMethod("vif")
```

```
> vif.default <- function(object, ...)
    stop("No default method for vif. Sorry.")

> vif.lm <- function(object, ...) {
    V <- summary(object)$cov.unscaled
    Vi <- crossprod(model.matrix(object))
    nam <- names(coef(object))
    k <- match("(Intercept)", nam,
      nomatch = FALSE)
    if(k) {
      v1 <- diag(V)[-k]
      v2 <- diag(Vi)[-k] - Vi[k, -k]^2 / Vi[k, k]
      nam <- nam[-k]
    }
    else {
      v1 <- diag(V)
      v2 <- diag(Vi)
      warning(paste("No intercept term",
        "detected. Results may surprise."))
    }
    structure(v1 * v2, names = nam)
  }
```

Uncentered variance inflation factors

As pointed out by Belsley (1991, p. 28/29) the centered VIF can be disadvantageous when collinearity effects are related to the intercept. To illustrate this point let us consider the following small ad-hoc example:

```
> n <- 50
> x2 <- 5 + rnorm(n, 0, 0.1)
> x3 <- 1:n
> x4 <- 10 + x2 + x3 + rnorm(n, 0, 10)
> x5 <- (x3/100)^5 * 100
> y <-  10 + 4 + x2 + 1*x3 + 2*x4 + 4*x5 +
      rnorm(n, 0, 50)
> cl.lm <- lm(y ~ x2 + x3 + x4 + x5)
```

Then centered variance inflation factors obtained via `vif.lm(cl.lm)` are

```
      x2       x3       x4       x5
1.017126 6.774374 4.329862 3.122899
```

Several runs produce similar results. Usually values greater than 10 are said to indicate harmful collinearity, which is not the case above, although the VIFs report moderate collinearity related to x_3 and x_4. Since the smallest possible VIF is 1, the above result seems to indicate that x_2 is completely unharmed by collinearity.

No harmful collinearity in the model when the second column $\boldsymbol{x}_2$ of the matrix $\boldsymbol{X}$ is nearly five times the first column and the scaled condition number, see Belsley (1991, Sec. 3.3), is 147.4653?

As a matter of fact, the centered VIF *requires an intercept in the model but at the same time denies the status of the intercept as an independent 'variable' being possibly related to collinearity effects.*

Now as indicated by Belsley (1991), an alternative way to measure variance inflation is simply to apply the classical (uncentered) coefficient of determination

$$R_j^2 = 1 - \frac{\boldsymbol{x}_j' \boldsymbol{M}_j \boldsymbol{x}_j}{\boldsymbol{x}_j' \boldsymbol{x}_j}$$

instead of the centered $\widetilde{R}_j^2$. This can also be done for the intercept as an independent 'variable'. For the above example we obtain

```
(Intercept)       x2       x3       x4       x5
  2540.378 2510.568 27.92701 24.89071 4.518498
```

as uncentered VIFs, revealing that the intercept and x_2 are strongly effected by collinearity, which is in fact the case.

It should be noted that $\boldsymbol{x}_j' \boldsymbol{C} \boldsymbol{x}_j \leq \boldsymbol{x}_j' \boldsymbol{x}_j$, so that always $\widetilde{R}_j^2 \leq R_j^2$. Hence the usual critical value $\widetilde{R}_j^2 > 0.9$ (corresponding to centered VIF > 10) for harmful collinearity should be greater when the uncentered VIFs are regarded.

Conclusion

When we use collinearity analysis for finding a possible relationship between impreciseness of estimation and near linear dependencies, then an intercept in the model should not be excluded from the analysis, since the corresponding parameter is as important as any other parameter (e.g. for prediction purposes). See also (Belsley, 1991, Sec. 6.3, 6.8) for a discussion of mean centering and the constant term in connection with collinearity.

By reporting both, centered and uncentered VIFs for a linear model, we can obtain an impression about the possible involvement of the intercept in collinearity effects. This can easily be done by complementing the above `vif.lm` function, since the uncentered VIFs can be computed as in the `else` part of the function, namely as `diag(V) * diag(Vi)`. By this, the uncentered VIFs are computed anyway and not only in the case that no intercept is in the model.

```
> vif.lm <- function(object, ...) {
    V <- summary(object)$cov.unscaled
    Vi <- crossprod(model.matrix(object))
    nam <- names(coef(object))
    k <- match("(Intercept)", nam,
      nomatch = FALSE)
    v1 <- diag(V)
    v2 <- diag(Vi)
    uc.struct <- structure(v1 * v2, names = nam)
    if(k) {
      v1 <- diag(V)[-k]
      v2 <- diag(Vi)[-k] - Vi[k, -k]^2 / Vi[k, k]
      nam <- nam[-k]
      c.struct <- structure(v1 * v2, names = nam)
      return(Centered.VIF = c.struct,
        Uncentered.VIF = uc.struct)
    }
```

```
    else{
      warning(paste("No intercept term",
        "detected. Uncentered VIFs computed."))
      return(Uncentered.VIF = uc.struct)
    }
  }
```

The user of such a function should allow greater critical values for the uncentered than for the centered VIFs.

Bibliography

D. A. Belsley (1991). *Conditioning Diagnostics. Collinearity and Weak Data in Regression*. Wiley, New York. 14

J. Fox (1997). *Applied Regression, Linear Models, and Related Methods*. Sage Publications, Thousand Oaks. 13

J. Fox (2002). *An R and S-Plus Companion to Applied Regression*. Sage Publications, Thousand Oaks. 13

W. N. Venables (2002). *[R] RE: [S] VIF Variance Inflation Factor*. http://www.R-project.org/nocvs/mail/r-help/2002/8566.html. 13

Jürgen Groß
University of Dortmund
Vogelpothsweg 87
D-44221 Dortmund, Germany
gross@statistik.uni-dortmund.de

Building Microsoft Windows Versions of R and R packages under Intel Linux

A Package Developer's Tool

by Jun Yan and A.J. Rossini

It is simple to build R and R packages for Microsoft Windows from an ix86 Linux machine. This is very useful to package developers who are familiar with Unix tools and feel that widespread dissemination of their work is important. The resulting R binaries and binary library packages require only a minimal amount of work to install under Microsoft Windows. While testing is always important for quality assurance and control, we have found that the procedure usually results in reliable packages. This document provides instructions for obtaining, installing, and using the cross-compilation tools.

These steps have been put into a Makefile, which accompanies this document, for automating the process. The Makefile is available from the contributed documentation area on Comprehensive R Archive Network (CRAN). The current version has been tested with R-1.7.0.

For the impatient and trusting, if a current version of Linux R is in the search path, then

```
make CrossCompileBuild
```

will run all Makefile targets described up to the section, *Building R Packages and Bundles*. This assumes: (1) you have the Makefile in a directory `RCrossBuild` (empty except for the makefile), and (2) that `./RCrossBuild` is your current working directory. After this, one should manually set up the packages of interest for building, though the makefile will still help with many important steps. We describe this in detail in the section on *Building R Packages and Bundles*.

Setting up the build area

We first create a separate directory for R cross-building, say `RCrossBuild` and will put the Makefile into this directory. From now on this directory is stored in the environment variable $RCB. Make sure the file name is `Makefile`, unless you are familiar with using the **make** program with other control file names.

Create work area

The following steps should take place in the `RCrossBuild` directory. Within this directory, the Makefile assumes the following subdirectories either exist or can be created, for use as described:

- `downloads` is the location to store all the sources needed for cross-building.
- `cross-tools` is the location to unpack the cross-building tools.
- `WinR` is the location to unpack the R source and cross-build R.
- `LinuxR` is the location to unpack the R source and build a fresh Linux R, which is only needed if your system doesn't have the current version of R.