```
######################################
##### Simple Regression Commands  ####
#####           2025              ####
######################################
```

```
# These are the basic commands for compiling Multiple Binary
Fixed-Effects Logistic Regression

# Packages
# There are two packages for doing a logistic regression
analysis.

# (MASS) -  comes pre-installed
# (rms) -   download / install

install.packages("rms", dependencies = TRUE)

# We use both the glm() function (package - MASS) + the lrm()
function (package rms).
# The glm() function output is simpler to read, but gives you
less information.

#load packages
library(MASS)
library(rms)
```

```
#################
##  Load Data  ##
#################
```

```
# Data format - make sure it is a flat data frame, not a
numerical summary / cross-tabulations of the results.
# Load the entire data frame, we will choose which variables
to use "in" R.

df <- read.table(file.choose(), header=T, sep="\t",
stringsAsFactors = T)
```

If you are using csv, then use

```
df <- read.table(file.choose(), header=T, sep=";",
stringsAsFactors = T)
```

Or

```
df <- read.table(file.choose(), header=T, sep=",",
```

```
stringsAsFactors = T)
```

```
####################
##  Examine Data  ##
####################

# To look at the data and check for spelling mistakes and to
make sure all has worked

summary(df)
str(df)


# Some of your variables will have more then 6 levels
(features / categories)
# R only lists the 7 most common features. To see the rest, to
check for spelling issues and small counts, use the function
table()
# In following line, replace "Name_of_Variable" with the name
of the variable (factor / column name) with more than 6 levels
# Copy and paste from the output of the summary() command
works well

table(df $ Name_of_Variable)

# Make sure you clean the data (spelling, small counts etc.)
in the original data - yr spreadsheet (e.g. Excel) or database
(e.g. Filemaker)
# When the data are 'clean', reload with the read.table()
function above.
```

```
####################################
#####  GLM Logistic Regression  #####
####################################

# Use the output from the summary() function to choose (and
copy and paste) the response variable and the predictor
variables
# Add all predictor variables you believe may have a role in
the outcome, separating each by a '+' sign.
# DV is your dependent variable, IVs are your independent
variables

library(MASS)

M = glm(DV ~ IV + IV + IV, data = df, family = "binomial")
summary(M)



# If you get an error message saying the model cannot
converge: there are two possible reasons
# -  you have a variable that has features with only a few
occurrences (12 is minimum, but aim for 20)
# -  you have two variables that have a feature which
co-occurs perfectly (predicts in the same way). Such variables
cannot be submitted simultaneously. See multicollinearity
below.



####################################
#####   LRM Logistic Regression   ####
####################################

# lrm does the same as glm, but it does not offer pretty stars
to show significance.
# lrm produces the model statistics, which allow you to
determine the predictive strength of the model

library(rms)

MM = lrm(DV ~ IV + IV + IV, data = df, x=T, y=T)
MM



###########################################
```

```
### Mixed Effects Logistic with mclogit ####
############################################

# mblogit does the same as glm and and lrm, but allows you to add random variables
# There are many packages that can be used for this, this is a good simple one.

library(mclogit)
library(survival)

MBLOG = mblogit(DV ~ IV +IV, ~ 1 | RV, data = df)
summary(MBLOG)
concordance(MBLOG)
```

```
#########################
##  Model Diagnostics  ##
#########################

## Parsimony ###

# You should already have a good idea if your model is
parsimonious based on your interpretation of the table of
coefficients, but it is worth checking that all is in order

anova(MM)


## Multicollinearity ###

# This concept is crucial to logistic regression and is
typically the biggest problem in corpus linguistics
# Logistic regression does NOT assume that the data is
normally distributed etc, but it does assume that a model is
'orthogonal'
# Each predictor variable must predict the outcome in a
different way. If you have two variables that are 'similar',
then this artificially improves the results (possible Type 1
error)
# Having no multicollinearity / having an orthogonal model is
essential

vif(MM)

# The vif() function calculates the 'variance inflation
factor'
# This is a figure that tells you if a given feature (level)
is collinear in the model.
# There is a debate over what is an acceptable level of
multicollinearity.
# Any number at or above 10 is clearly a problem
# Any number above 4 should be reported, as many (if not most)
people consider 4 to be the upper-limit of acceptability.
# Some people consider 2.5 to be the maximum VIF.
# This command should work on both glm and lrm models
```

Bootstrapping

Bootstrapping is a randomisation method which allows you to better estimate discriminatory power and significance.
We can apply it to the model to 'check' the model statistics, such as the Pseudo R2 and Concordance statistic

```
validate(MM, bw = T, B = 200)
```

If the bootstrapped R2 or C is considerably lower than that produced with the lrm() function, then the model may have problems and its predictive strength should be interpreted with great care.

To calculate the bootstrapped C score, use the below equation. Replace Dxy with the bootstrapped Sommers D_{xy}

```
(Dxy / 2) + 0.5
```

Outlyers

Outliers and influential observations can have a large impact on your model, both positive and negative. The best way to find them is to plot your model. Apply this function to the model produced with glm nor lrm.

```
plot(M)
```

Referent Level

In the table of coefficients, each predictor is calculated relative to one of the levels. If you change that “referent” level, the table of coefficients may change considerably.

Note that although the effect sizes and behaviour in the table of coefficients may change, the performance of the overall model is not affected.

```
df $ Name_of_Variable = relevel(df $ Name_of_Variable, ref=
"New_Referent_Level_Name")
```

Don’t forget to re-run the model before you ask for the summary after you have changed the referent level

Interactions

Interactions are an important part of predictive modelling. To add an interaction replace "+" with "*"

```
M = glm(DV ~ IV * IV, data = df, family = "binomial")
summary(M)
```

Interactions require a great deal of data and so often cannot be used