UNIVERSITEIT VAN DIE VRYSTAAT
UNIVERSITY OF THE FREE STATE

STSB 6816

WISKUNDIGE STATISTIEK & AKTUARIËLE WETENSKAP/
MATHEMATICAL STATISTICS & ACTUARIAL SCIENCE

Test 2 — 23 July 2020

MEMORANDUM

TYD/TIME: 1800 Minutes PUNTE/MARKS: 100

INSTRUCTIONS:

- Answer all questions in a single R Markdown document. Knit to Word at the end and submit both for assessment in a single submission. Alternatively, you may work directly in Word at a penalty of 2% of the total test mark allocation.
- Label questions clearly using headings, as it is done on this question paper. Verify that you have done this correctly by noting the structure in the Word Navigation Pane.
- All results accurate to 2 decimal places.
- Show all derivations, formulas, code, sources and reasoning. Sources must be cited in text and referenced at the end, with links to websites where information or code was found.
- Intervals should cover 95% probability unless stated otherwise, and hypothesis tests should assume $\alpha = 0.05$.
- No communication software, no communication devices, and no communication capable websites may be accessed prior to submission, except for the purpose of posing questions to the lecturer. You may not (nor even appear to) attempt to communicate or pass information to another student. Thus, your final submission must be truly unique to you in all respects.

Question 1

The Generalised Pareto Distribution (GPD) is used to model extreme observations above a threshold or cut-off point. One situation where it is commonly used is reinsurance claims. The density is

$$f_X(x) = \sigma^{-1}\left[1 + \frac{\xi(x-\mu)}{\sigma}\right]^{-\xi^{-1}-1}$$

The key parameter of the GPD is the extreme value index (EVI), which we will denote ξ (xi), that measures the extremeness of the extreme values. This parameter is generally assumed constant but unknown for a specific data generating process.

Another parameter is the threshold (μ) which we will assume to be 0 for this problem. In any situation where the threshold value is known it can be subtracted from all larger observations and then assumed to be 0. In reinsurance contracts the threshold is usually specified in writing and is thus known.

The last parameter is the scale parameter σ, this parameter stretches out the distribution to fit the scale of the data, and must be estimated carefully as it is negatively correlated with ξ. The problem is that the scale can easily change over time due to many possible factors. One possible factor is inflation.

You are provided a set of extreme claims with the threshold already subtracted and set to 0. You are also provided with an inflation index matching the time of each claim. The claim times are given as days since the introduction of this type of policy on 1 July 2014.

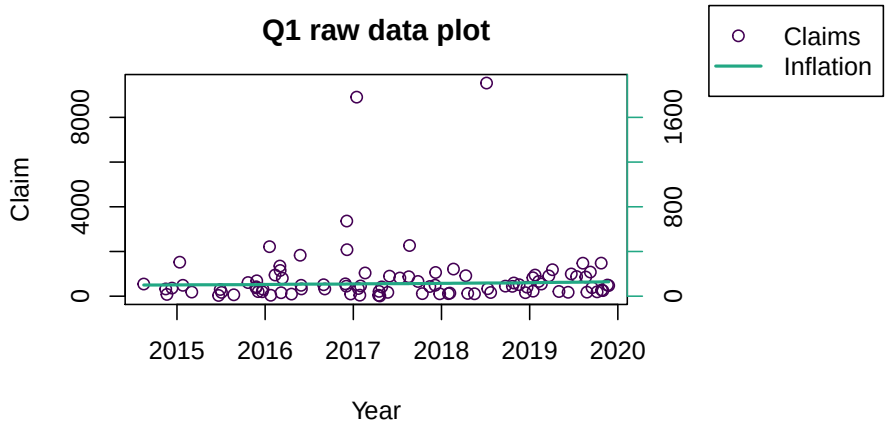
It is your task to fit two GPD distribution models to the claims:

Model 1: Assume that the scale parameter grows with inflation ($\sigma_i = \sigma * Index_i$), so that the observations can be divided by the index to arrive at an i.i.d. GPD sample with a fixed scale and fixed EVI. Assume a positive EVI to avoid boundary issues. Use the MDI prior for the GPD $\propto \sigma^{-1} \exp(-\xi)$ or $c_1 - \log\sigma - \xi$ on the log scale (c_1 being an unknown constant), or $c_2 + \log\alpha - \log\beta - \alpha$ if you express the GPD as a gamma mixture of exponential random variables.

Model 2: Ignore the inflation index entirely and instead assume that the scale parameter increases gradually over time in a gradual exponential curve, *i.e.* σ_i (*or* β_i) $= \exp(\sigma_0 + \delta * t_i)$ where t_i is the time since the start of the contract (also given in your data). This model should take into account that δ must be positive. Again assume a positive EVI to avoid boundary issues, perhaps with an Exp(1) prior or similar.

(a) Load the data corresponding to your student number and draw a plot of claims (as points) and inflation (as a line) versus time. [4]

```
library(openxlsx)
mydata <- read.xlsx('STSB6816Test2of2020Q1.xlsx',paste0('St__',st))
names(mydata)
y <- mydata$Claims
days <- mydata$DaysFromStart
year <- mydata$DaysFromStart/365.25 + 2014.5
infl <- mydata$Index

library(viridisLite)
cols <- viridis(6)
par(mar=c(4.4,4.4,3,9))
plot(year, y, main='Q1 raw data plot', xlab = 'Year', ylab = 'Claim', col=cols[1])
places <- pretty(range(y))
axis(4, at=places, labels = as.character(round(places/5)), col = cols[4])
lines(year, infl*5, col=cols[4], lwd=2)
legend('topright', c('Claims','Inflation'), pch=c(1,NA), lwd=c(NA,2), lty=c(NA,1),
    col=cols[c(1,4)], inset = c(-0.5,-0.3), xpd=NA)
```



✔✔✔✔

(b) For Model 1, implement the model in any way you choose, as best you can. Illustrate the joint posterior distribution of the two parameters on a graph of your choosing. [10]

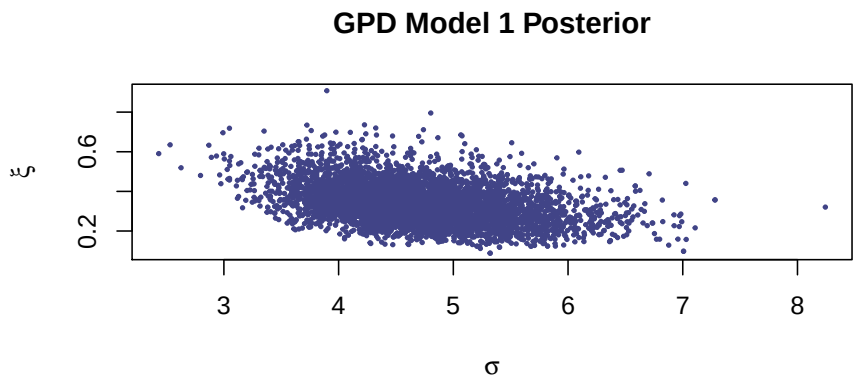
[Remember to divide the claims by the inflation index first then fit the GPD as if you had an i.i.d. sample.]

```
library(parallel)
library(rstan)
mycores <- max(1,floor(detectCores(logical = FALSE)*0.75))
options(mc.cores = mycores)
rstan_options(auto_write = TRUE)
library(loo)
```

```
// This Stan block defines a Generalised Pareto Distribution model with MDI prior, by
     Sean van der Merwe, UFS
data {
  int<lower=0> n;                 // number of observations
  real<lower=0> y[n];                     // observations
}
// The parameters of the model
parameters {
  real<lower=0> a;             // tail index
  real<lower=0> b;                  // scale parameter
  real<lower=0> lambda[n];     // mixing variable
}
// The model to be estimated.
model {
  y ~ exponential(lambda);   // GPD as mixture
  lambda ~ gamma(a,b);
  target += log(a)-log(b)-a;  // MDI prior
}
generated quantities {
  vector[n] log_lik;
  for (i in 1:n) {
    log_lik[i] = exponential_lpdf(y[i] | lambda[i]);
  }
}
```

✔✔✔✔✔✔

```
n <- length(y)
yscld <- y/infl
stan_data <- list(n=n, y=yscld)
ModelFit1 <- sampling(GPD1, stan_data, pars=c('a','b','log_lik'), iter = 10000,
    chains = mycores, algorithm='HMC')
draws1 <- extract(ModelFit1)
summary(ModelFit1)$summary[1:2,]

xi1 <- 1/draws1$a
sigma1 <- draws1$b/draws1$a
plot(sigma1[1:5000], xi1[1:5000], col=cols[2], pch=20, cex=0.5, main='GPD Model 1
    Posterior', xlab=expression(sigma), ylab=expression(xi))
```

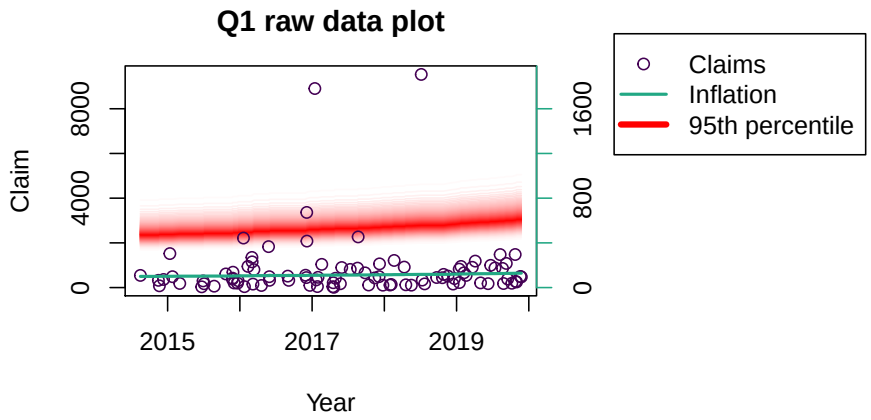


✔✔ ✔✔

(c) Redraw your data plot but now include the 95th upper percentile suggested by

Model 1 as an additional line. Explain how parameter uncertainty was handled in deciding where to draw the line, or draw fuzzy/additional lines that illustrate the parameter uncertainty.
Note that the quantile function of the GPD is $[(1-p)^{-\xi}-1]\sigma/\xi$. [6]

```
p95 <- (0.05^(-xi1)-1)*draws1$b

fp <- seq(0.01,0.99,0.01)
par(mar=c(4.4,4.4,3,9))
plot(year, y, main='Q1 raw data plot', xlab = 'Year', ylab = 'Claim', type='n')
for (i in 1:length(fp)) {
  yi <- quantile(p95,fp[i])*infl
  lines(year, yi, col=rgb(1, abs(0.5-fp[i])*2, abs(0.5-fp[i])*2))
}
points(year, y, col=cols[1])
places <- pretty(range(y))
axis(4, at=places, labels = as.character(round(places/5)), col = cols[4])
lines(year, infl*5, col=cols[4], lwd=2)
legend('topright', c('Claims','Inflation','95th percentile'), pch=c(1,NA,NA), lwd=c(
    NA,2,4), lty=c(NA,1,1), col=c(cols[c(1,4)],rgb(1,0,0)), inset = c(-0.5,-0.14),
    xpd=NA, cex=0.6)
```



✔✔ ✔✔

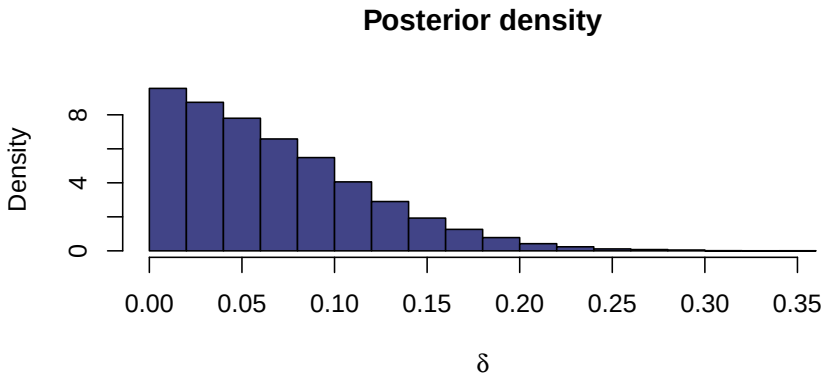
Uncertainty was included in the percentile line either by adding an interval to the line or by making the line fuzzy according to the quantiles. ✔✔

(d) Now revert back to the original data set and fit Model 2 as best you can. Illustrate the posterior distribution of the slope parameter (δ). [11]

```
// This Stan block defines a Generalised Pareto Distribution model with growth curve,
     by Sean van der Merwe, UFS
data {
  int<lower=0> n;                 // number of observations
  real<lower=0> y[n];                    // observations
  vector[n] ti;                    // timestamps
}
// The parameters of the model
parameters {
  real<lower=0> a;             // tail index
  real<lower=0> s0;                 // scale parameter
  real<lower=0> d;                  // scale parameter
  real<lower=0> lambda[n];    // mixing variable

}
// The model to be estimated.
model {
  y ~ exponential(lambda);   // GPD as mixture
  lambda ~ gamma(a, exp(d*ti+s0));
  a ~ exponential(1);  //  prior
  d ~ exponential(10);  //  prior
  s0 ~ normal(6,10);  //  prior
}
generated quantities {
  vector[n] log_lik;
  for (i in 1:n) {
    log_lik[i] = exponential_lpdf(y[i] | lambda[i]);
  }
}
```

✔✔✔✔✔✔

```
n <- length(y)
ti <- days/365.25
stan_data <- list(n=n, y=y, ti=ti)
ModelFit2 <- sampling(GPD2, stan_data, pars=c('a','d','s0','log_lik'), iter = 10000,
    chains = mycores, control=list(adapt_delta=0.95))
draws2 <- extract(ModelFit2)
summary(ModelFit2)$summary[1:3,]

hist(draws2$d, main='Posterior density', freq=FALSE, xlab = expression(delta), col=
    cols[2])
```

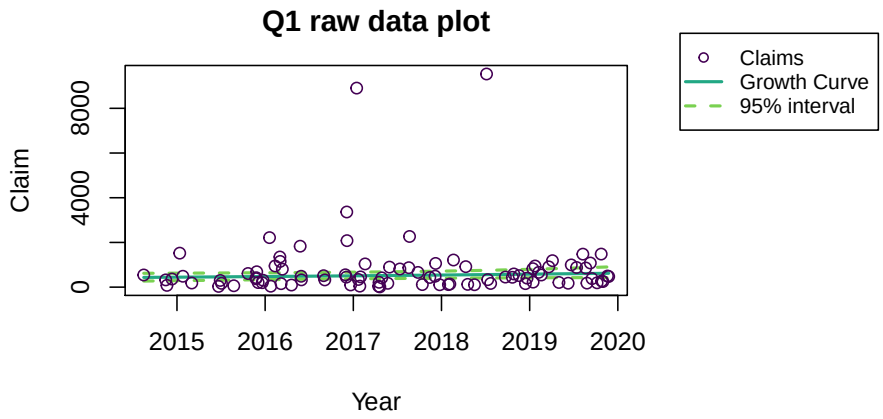


✔✔✔✔ ✔

(e) Redraw your original data plot but now replace inflation with the growth curve suggested by Model 2. Be sure to also illustrate the uncertainty in your estimation of the growth curve. [6]

```
xi2 <- 1/draws2$a
beta2 <- sapply(ti, function(yrs) { exp(draws2$d*yrs+draws2$s0) } )
sigma2 <- beta2*matrix(xi2,length(xi2),n)

par(mar=c(4.4,4.4,3,9))
plot(year, y, main='Q1 raw data plot', xlab = 'Year', ylab = 'Claim', type='n')
lines(year, colMeans(sigma2), col=cols[4], lwd=2)
lines(year, apply(sigma2,2,quantile,0.025), col=cols[5], lwd=2, lty=2)
lines(year, apply(sigma2,2,quantile,0.975), col=cols[5], lwd=2, lty=2)
points(year, y, col=cols[1])
legend('topright', c('Claims','Growth Curve','95% interval'), pch=c(1,NA,NA), lwd=c(
    NA,2,2), lty=c(NA,1,2), col=cols[c(1,4,5)], inset = c(-0.45,-0.14), xpd=NA, cex
    =0.8)
```

✔✔✔✔



✔✔

(f) Compare Model 1 to Model 2 considering generalisability or parsimony. [4]

```
log_lik_1 <- extract_log_lik(ModelFit1, merge_chains = FALSE)
log_lik_2 <- extract_log_lik(ModelFit2, merge_chains = FALSE)
r_eff_1 <- relative_eff(exp(log_lik_1), cores = mycores)
r_eff_2 <- relative_eff(exp(log_lik_2), cores = mycores)
loo_1 <- loo(log_lik_1, r_eff = r_eff_1, cores = mycores)
loo_2 <- loo(log_lik_2, r_eff = r_eff_2, cores = mycores)
comp <- loo_compare(loo_1, loo_2)
print(comp, simplify=FALSE)
```

✔✔

✔

Model	elpd diff	se diff	elpd loo	p loo	se ploo	looic	se looic
model1	0	0	-286.6	12.8	2	573.2	20
model2	-461.8	0.9	-748.4	14.2	2.1	1496.8	20.2

Model 1 is a far more parsimonious fit because it explains the data better using less parameters. ✔

Total for Question 1: 41

Question 2

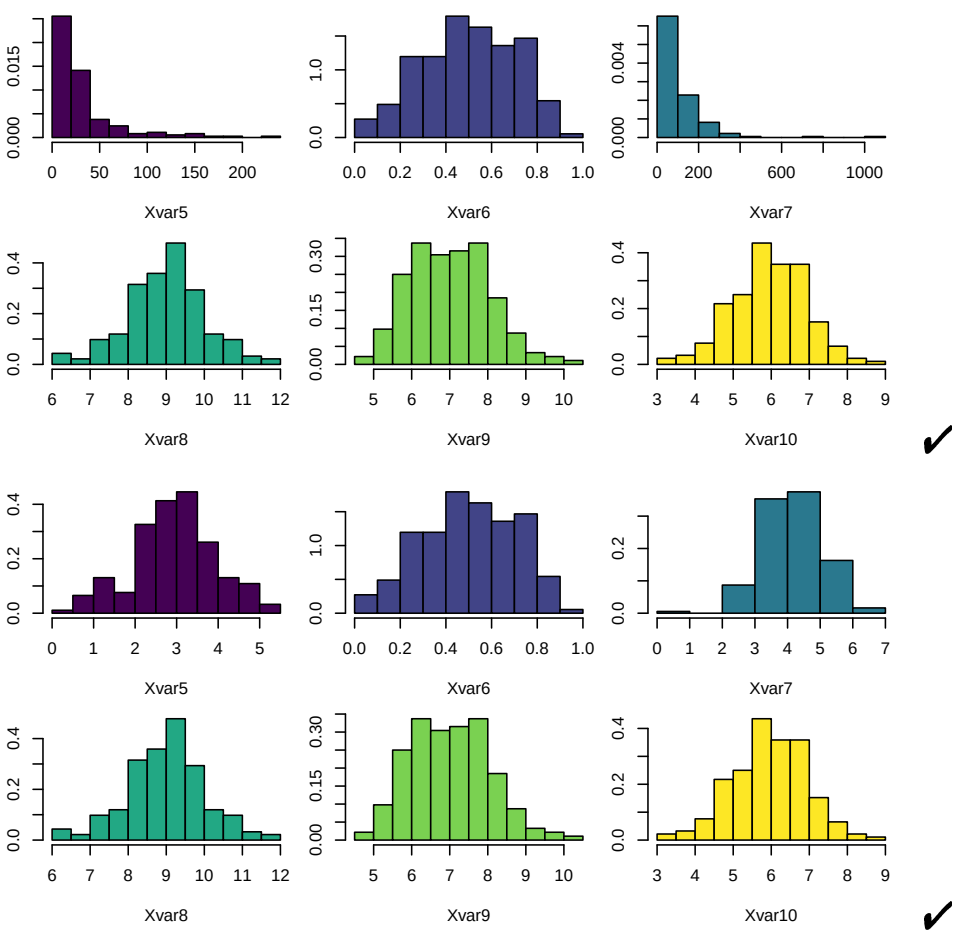
In the file provided you will find a set of data on the sheet corresponding to your student number. The data has many variables of different types. You are tasked with building appropriate regression models for the first two of those variables, dependent on all the others, and interpreting the results.

The first variable is an objective measure of subject illness on an arbitrary real scale. The second variable is a subjective measure of subject illness on a scale of 0 to [maximum in your data]. You also have available the subject age, various measures that might explain subject illness, and the subject ID.

(a) Begin by drawing small histograms of all numeric *Xvar* variables to see whether any have excessive skewness. If so, take the natural log of those variables and draw new histograms to see whether the impact was good. Explain why this step is useful. [8]

```
mydata2 <- read.xlsx('STSB6816Test2of2020Q2.xlsx' ,paste0('St__',st))
(vnames <- names(mydata2))

nxvar <- ncol(mydata2)-4
Sbj <- factor(mydata2$Subject)
nSbj <- nlevels(Sbj)
SbjNum <- as.numeric(Sbj)

# Function to calculate sample skewness, by Sean van der Merwe, UFS
skewness <- function(x) {
(x - mean(x)) -> stdx
s0 <- mean(stdx^3)/(( mean(stdx^2) )^1.5)
n <- length(x)
s1 <- s0*sqrt(n*(n-1))/(n-2)
return(s1) }

par(mfrow=c(2,3), mar=c(4.4,2.4,0.4,0.4))
for (i in 1:nxvar) {
  hist(mydata2[[i+3]], xlab = vnames[i+3], main='', freq=FALSE, col = cols[i])
}
skw <- lapply(mydata2[4:(3+nxvar)],skewness)

for (i in 1:nxvar) {
  if (skw[i] > 1) {
    mydata2[[i+3]] <- log(mydata2[[i+3]])
  }
}

par(mfrow=c(2,3), mar=c(4.4,2.4,0.4,0.4))
for (i in 1:nxvar) {
  hist(mydata2[[i+3]], xlab = vnames[i+3], main='', freq=FALSE, col = cols[i])
}
```

✔ ✔ ✔ ✔



When explanatory variables are excessively skew then a regression model will focus on the extreme values instead of capturing the relationships in the bulk of the data as usually desired. ✔ ✔

(b) Regress the first variable on all the others except the second variable (as linear main effects only), assuming a t density for the dependent variable conditional on the explanatory model. Discuss which explanatory variables, other than Subject(s), can be considered statistically significant. [Tip: use as prior the assumption that the degrees of freedom parameter ν (nu) follows an exponential(0.1) density.] [12]

```
data {
  int<lower=0> n;                  // number of observations
  vector[n] y;               // observations
  int<lower=0> nx;     // number of real explanatory variables
  matrix[n,nx] X;               // Explanatory variables
  int<lower=0> ns;     // number of subjects
  int<lower=0,upper=ns> s[n];     // subject IDs
}
// The parameters of the model
parameters {
  vector[nx] slopes;                    // slopes of explanatory variables
  real intercepts[ns];     // Subject effect
  real<lower=0> sigma;  // standard deviation of errors
  real<lower=0> nu; // degrees of freedom
}
transformed parameters {
  vector[n] mu;                    // the expected values (linear predictor)
  for (i in 1:n) {                 // every observation has a different model
    mu[i] = intercepts[s[i]];           // subject effects
  }
}
// The model to be estimated.
model {
    y ~ student_t(nu, mu + X*slopes, sigma);
    nu ~ exponential(0.1);
}
```

✔✔✔✔

```
stan_data_t1 <- list(y=mydata2$Measurement1, n=nrow(mydata2), X=mydata2[,3:(3+nxvar)
    ], nx=nxvar+1, ns=nSbj, s=SbjNum)
modelfit_t1 <- sampling(tmodel, stan_data_t1, pars=c('intercepts', 'slopes', 'sigma',
     'nu'), iter = 10000, chains = mycores)
list_of_draws_t1 <- extract(modelfit_t1)

# summary(modelfit_t1)$summary
pvalfunc <- function(sims,target=0) { 2*min(mean(sims<target),mean(sims>target)) }
sigfunc <- function(pvalue) { c('significant','not significant')[c(pvalue<=alpha,
    pvalue>alpha)] }
alpha <- 0.05

estimates_t1 <- apply(list_of_draws_t1$slopes, 2, mean)
pvals_t1 <- apply(list_of_draws_t1$slopes, 2, pvalfunc)
write.xlsx(data.frame(Variable=vnames[3:(3+nxvar)], Estimate=round(estimates_t1,2), p
    _value=round(pvals_t1,2), Significance=sapply(pvals_t1,sigfunc)), 'TempTable.xlsx
    ')
```

✔✔✔✔✔

Variable	Estimate	p-value	Significance
Age	0.49	0.03	significant
Xvar5	1.59	0.17	not significant
Xvar6	0.70	0.54	not significant
Xvar7	-0.59	0.64	not significant
Xvar8	4.93	0.41	not significant
Xvar9	-73.83	0.00	significant

Either look at which intervals don't cover 0, or use a Bayesian approximation of p-value to check which terms are significant. ✔✔✔

(c) Explain when and why one might want to use a t density instead of a Gaussian

density for a dependent variable. [3]

The t density can be expressed as a mixture of normal densities, so that a t regression can accommodate random heteroscedasticity ✔— so individual observations having different variances in a non-systematic way. It also accommodates heavier tails ✔and is more robust to outliers ✔.

(d) Calculate 90% prediction intervals for the values of your dependent variable and calculate the empirical coverage of your model. Give only the empirical coverage and attempt to explain any deviation from 90% coverage. [5]

```
shortestinterval <- function(postsims,alpha=0.05) { # Coded by Sean van der Merwe,
    UFS
sorted.postsims <- sort(postsims)
nsims <- length(postsims)
gap <- round(nsims*(1-alpha))
widths <- diff(sorted.postsims,gap)
interval <- sorted.postsims[c(which.min(widths),(which.min(widths) + gap))]
return(interval) }

ny <- nrow(mydata2)
nsims <- length(list_of_draws_t1$nu)
inint2 <- inint <- rep(TRUE,ny)
Xmat <- as.matrix(mydata2[,3:(3+nxvar)])
for (i in 1:ny) {
  preds <- rt(nsims,list_of_draws_t1$nu)*list_of_draws_t1$sigma + list_of_draws_t1$
      intercepts[,SbjNum[i]] + c(list_of_draws_t1$slopes%*%Xmat[i,])
  quants <- shortestinterval(preds,0.1)
  quants2 <- quantile(preds,c(0.05,0.95))
  inint[i] <- ((quants[1] < mydata2$Measurement1[i]) & (quants[2] > mydata2$
      Measurement1[i]))
  inint2[i] <- ((quants2[1] < mydata2$Measurement1[i]) & (quants2[2] > mydata2$
      Measurement1[i]))
}
(coverage <- mean(inint))
(coverage2 <- mean(inint2))
```

✔✔✔

Coverage numbers close to 0.9 indicate good coverage. A little bit below may indicate a failure to capture all variation, while a little bit over could indicate that the t density was unnecessary. However, there is always some simulation error so one should not read into small variations too much. ✔✔

(e) Adapt your model to make the Subject variable (last column) a random effect grouping variable instead of a fixed effect as in the previous model. Report the estimated standard deviation of the random effect as a proportion of the estimated standard deviation of the residuals (point estimates are fine here). [8]

```
data {
  int<lower=0> n;                  // number of observations
  vector[n] y;               // observations
  int<lower=0> nx;     // number of real explanatory variables
  matrix[n,nx] X;               // Explanatory variables
  int<lower=0> ns;     // number of subjects
  int<lower=0,upper=ns> s[n];     // subject IDs
}
// The parameters of the model
parameters {
  vector[nx] slopes;                    // slopes of explanatory variables
  real intercepts[ns];     // Subject effect
  real intercept;    // general intercept
  real<lower=0> sigma;  // standard deviation of errors
  real<lower=0> nu; // degrees of freedom
  real<lower=0> sSbj;  // standard deviation of subject effects
}
transformed parameters {
  vector[n] mu;                    // the expected values (linear predictor)
  for (i in 1:n) {                 // every observation has a different model
    mu[i] = intercepts[s[i]];           // subject effects
  }
}
// The model to be estimated.
model {
    y ~ student_t(nu, mu + X*slopes, sigma);
    intercepts ~ normal(intercept, sSbj);
    nu ~ exponential(0.1);
    sSbj ~ exponential(0.1);
}
```

✔ ✔ ✔ ✔

```
stan_data_t2 <- list(y=mydata2$Measurement1, n=nrow(mydata2), X=mydata2[,3:(3+nxvar)
    ], nx=nxvar+1, ns=nSbj, s=SbjNum)
modelfit_t2 <- sampling(tmodel2, stan_data_t2, pars=c('intercepts', 'slopes', 'sigma'
    , 'nu', 'intercept','sSbj'), iter = 10000, chains = mycores, control=list(adapt_
    delta=0.99))
list_of_draws_t2 <- extract(modelfit_t2)

summary(list_of_draws_t2$sSbj/list_of_draws_t2$sigma)
```

✔ ✔ ✔

The proportion should be not too small, probably around 0.25. ✔

(f) When is it appropriate to drop insignificant explanatory variables and refit the model, and when should a variable not be dropped even if it appears insignificant? [3]

Dropping variables that are effectively noise in the model will allow the model to better focus on the variables that matter and give more stable estimates of parameters. However, dropping variables that are at the core focus of the research means that the regression is useless no matter how well it fits. ✔ ✔ ✔

(g) Adapt your model to use the prior density given below. Illustrate the posterior density of ν that results. [5]

$$p(\nu) \propto \left[\psi'(0.5\nu) - \psi'(0.5(\nu+1)) - 2(\nu+3)\nu^{-1}(\nu+1)^{-2}\right]^{0.5}$$

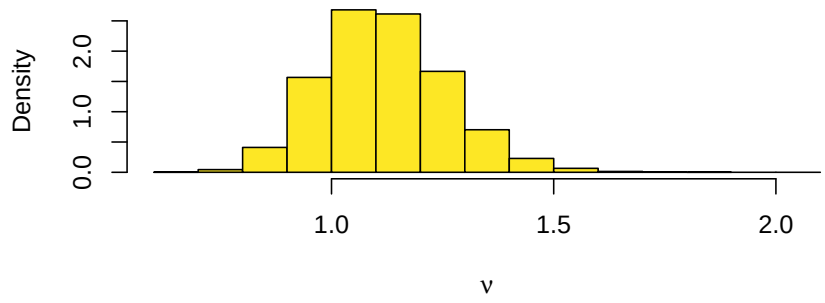
where ψ' is the trigamma function.

```
...
// The model to be estimated.
model {
    y ~ student_t(nu, mu + X*slopes, sigma);
    intercepts ~ normal(intercept, sSbj);
    sSbj ~ exponential(0.1);
    target += 0.5*log(trigamma(0.5*nu) - trigamma((nu+1)*0.5) - 2*(nu+3)/nu/(nu+1)/(
        nu+1));
}
```

✔ ✔

```
stan_data_t3 <- list(y=mydata2$Measurement1, n=nrow(mydata2), X=mydata2[,3:(3+nxvar)
    ], nx=nxvar+1, ns=nSbj, s=SbjNum)
modelfit_t3 <- sampling(tmodel3, stan_data_t3, pars=c('intercepts', 'slopes', 'sigma'
    , 'nu', 'intercept','sSbj'), iter = 10000, chains = mycores, control=list(adapt_
    delta=0.99, max_treedepth=12))
list_of_draws_t3 <- extract(modelfit_t3)

hist(list_of_draws_t3$nu, xlab = expression(nu), main='', freq=FALSE, col = cols[6])
```

✔ ✔



✔

(h) Regress the second variable on all the others except the first variable (as linear main effects only), assuming an appropriate density of your choosing for the dependent variable conditional on the explanatory model. Estimate the coefficients of the explanatory variables (except Subject) and interpret the values of those coefficients which appear significant. Also give the probability that the Age coefficient is larger than the others. [15]

```
data {
  int<lower=0> n;                  // number of observations
  int<lower=0> m;    // maximum integer value of y
  int<lower=0,upper=m> y[n];              // observations
  int<lower=0> nx;    // number of real explanatory variables
  matrix[n,nx] X;               // Explanatory variables

}
// The parameters of the model
parameters {
  vector[nx] slopes;                  // slopes of explanatory variables
}
transformed parameters {
  real mu[n];                    // the expected values (linear predictor)
  for (i in 1:n) {                 // every observation has a different model
    mu[i] = X[i,]*slopes;
  }
}
// The model to be estimated.
model {
    y ~ binomial_logit(m, mu);
}
```

✔✔✔✔✔✔

```
Xbin <- model.matrix(Response2~.-Measurement1, data=mydata2)

stan_data_bin <- list(y=mydata2$Response2, n=nrow(mydata2), X=Xbin, nx=ncol(Xbin), m=
    max(mydata2$Response2))
modelfit_bin <- sampling(binmodel, stan_data_bin, pars=c('slopes'), iter = 10000,
    chains = mycores)
list_of_draws_bin <- extract(modelfit_bin)

estimates_bin <- apply(list_of_draws_bin$slopes[,2:(2+nxvar)], 2, mean)
pvals_bin <- apply(list_of_draws_bin$slopes[,2:(2+nxvar)], 2, pvalfunc)
larger_bin <- apply(list_of_draws_bin$slopes[,2:(2+nxvar)], 2, function(x) { mean(
    list_of_draws_bin$slopes[,2] > x) })
write.xlsx(data.frame(Variable=vnames[3:(3+nxvar)], Estimate=round(estimates_bin,3),
    p_value=round(pvals_bin,3), Significance=sapply(pvals_bin,sigfunc), AgeCoefLarger
    =round(larger_bin,3)), 'TempTable2.xlsx')
```

✔✔✔✔

✔

Variable	Estimate	p-value	Significance	AgeCoefLarger
Age	0.004	0.69	not significant	0
Xvar5	-0.057	0.276	not significant	0.872
Xvar6	0.008	0.859	not significant	0.459
Xvar7	-0.019	0.737	not significant	0.654
Xvar8	-0.244	0.338	not significant	0.834
Xvar9	-0.805	0	significant	1

Interpretation of significant coefficient values in terms of log odds, or better. ✔✔ ✔✔

Total for Question 2: 59

Total half marks on memo = 200 vs. 200 = Double total margin points (=100).