import.sas

```
/* ----------------------------------------------------------------------------
   Instructions:
   1. Download the `games.csv` file from the Git repository.
   2. Upload it manually into SAS Studio (right-click > Upload).
   3. Go to the Files panel > right-click on the file > Properties > copy the path.
   4. Paste this path into the "pathfile" macro variable below.
   ---------------------------------------------------------------------------- */

/* TO DO: modify the path below with the path found in Properties */
%let pathfile = '/home/u64112561/ml_steam/data/games.csv';

proc import datafile = "&pathfile"
            dbms = csv
            out = steam_games
            replace;
            guessingrows = 4000;
run;
```

setup.sas

```
/*  run this script once per session*/

DATA GAMES;
set STEAM_GAMES(rename=(
            'Average playtime forever'n = Average_playtime_forever
            'Peak CCU'n = Peak_CCU
            'Required age'n = Required_age
```

```
        'Estimated owners'n = estimated_owners
        ));
        keep Average_playtime_forever Peak_CCU
        Price Recommendations Required_age
        Positive Negative total_reviews Positive_Ratio rating_levels Estimated_owners ;
 length estimated_owners $10;

Total_Reviews = Positive + Negative;
Positive_Ratio = Positive  / Total_Reviews;
Average_Review = (Positive / (Positive + Negative));
Supported_Languages = countw('Supported Languages'n);
Release_Year = year(Release_Date);

if Average_Playtime_Forever > 0 ;


if total_reviews > 0 then
        positive_ratio = (Positive / total_reviews) * 100;
        else
        positive_ratio = .;

        length rating_levels $30;

        if total_reviews >= 500 then do;
        if 95 <= positive_ratio <= 100 then rating_levels = "Overwhelmingly Positive";
        else if 80 <= positive_ratio < 95 then rating_levels = "Very Positive";
        else if 70 <= positive_ratio < 80 then rating_levels = "Mostly Positive";
        else if 40 <= positive_ratio < 70 then rating_levels = "Mixed";
        else if 20 <= positive_ratio < 40 then rating_levels = "Mostly Negative";
        else if 0 <= positive_ratio < 20 then rating_levels = "Overwhelmingly Negative";
        end;
```

```
        else if 50 <= total_reviews < 500 then do;
        if 80 <= positive_ratio <= 100 then rating_levels = "Very Positive";
        else if 70 <= positive_ratio < 80 then rating_levels = "Mostly Positive";
        else if 40 <= positive_ratio < 70 then rating_levels = "Mixed";
        else if 20 <= positive_ratio < 40 then rating_levels = "Mostly Negative";
        else if 0 <= positive_ratio < 20 then rating_levels = "Very Negative";
        end;
        else if 10 <= total_reviews < 50 then do;
        if 80 <= positive_ratio <= 100 then rating_levels = "Positive";
        else if 70 <= positive_ratio < 80 then rating_levels = "Mostly Positive";
        else if 40 <= positive_ratio < 70 then rating_levels = "Mixed";
        else if 20 <= positive_ratio < 40 then rating_levels = "Mostly Negative";
        else if 0 <= positive_ratio < 20 then rating_levels = "Negative";
        end;
        else rating_levels = "Not enough reviews";



 if Estimated_owners = "0 - 20000" then estimated_owners = "0-20k";
        else if Estimated_owners = "20000 - 50000" then estimated_owners = "20k-50k";
        else if Estimated_owners = "50000 - 100000" then estimated_owners = "50k-100k";
        else if Estimated_owners = "100000 - 200000" then estimated_owners = "100k-200k";
        else if Estimated_owners = "200000 - 500000" then estimated_owners = "200k-500k";
        else if Estimated_owners = "500000 - 1000000" then estimated_owners = "500k-1M";
        else if Estimated_owners = "1000000 - 2000000" then estimated_owners = "1M-2M";
        else if Estimated_owners = "2000000 - 5000000" then estimated_owners = "2M-5M";
        else if Estimated_owners = "5000000 - 10000000" then estimated_owners =
"5M-10M";
        else if Estimated_owners = "10000000 - 20000000" then estimated_owners =
"10M-20M";
```

```
        else if Estimated_owners = "20000000 - 50000000" then estimated_owners =
"20M-50M";
        else if Estimated_owners = "50000000 - 100000000" then estimated_owners =
"50M-100M";
        else if Estimated_owners = "100000000 - 200000000" then estimated_owners =
"100M-200M";
run;
```

unvariate.sas

```
data univariate;
        set GAMES;
        keep Average_playtime_forever Peak_CCU Price Recommendations Required_age
        Positive Negative total_reviews Positive_Ratio;
run;

proc univariate data=univariate;
        histogram _All_;
run;

%let varlist=Average_playtime_forever Peak_CCU
        Price Recommendations Required_age
        Positive Negative total_reviews Positive_Ratio;

%macro plot_vars;
        %let i = 1;
        %let var = %scan(&varlist, &i, %str( ));

        %do %while(%length(&var) > 0);

        proc plot data=univariate hpercent=50;
        plot &var * Average_playtime_forever / vspace=1;
```

```
        title "Plot de &var en fonction de rating";
        run;
        %let i = %eval(&i + 1);
        %let var = %scan(&varlist, &i, %str( ));
        %end;
        %mend;

        %plot_vars;
```

unvariatelog10.sas

```
ods graphics on;
/* Step 1: Create log-transformed variables */

data univariatelog10;
        set GAMES;
        log_Average_playtime_forever=log10(Average_playtime_forever + 1);
        log_Peak_CCU=log10(Peak_CCU + 1);
        log_Positive=log10(Positive + 1);
        log_Negative=log10(Negative + 1);
        log_Recommendations=log10(Recommendations + 1);
        log_Price=log10(Price + 1);
run;

/* Step 2: Define variables to plot */

%let varlist = log_Peak_CCU log_Positive log_Negative log_Recommendations log_Price;

/* Step 3: Define macro to plot each variable using SGPLOT */

%macro plot_vars;
        %let i = 1;
```

```
        %let var = %scan(&varlist, &i, %str( ));

        %do %while(%length(&var) > 0);
        title "Plot of &var against log_Average_playtime_forever";

        proc sgplot data=univariatelog10;
        scatter x=&var y=log_Average_playtime_forever;
        reg x=&var y=log_Average_playtime_forever / lineattrs=(color=red);
        run;

        %let i = %eval(&i + 1);
        %let var = %scan(&varlist, &i, %str( ));
        %end;
%mend;
/* Step 4: Run the macro */

%plot_vars;
ods graphics off;
```

correlation_matrix.sas

```
/* Step 1: Compute Spearman correlation matrix */
proc corr data=GAMES spearman nosimple noprint outp=corr_out;
        var _numeric_;
run;

/* Step 2: Sort before transpose */
proc sort data=corr_out(where=(_TYPE_="CORR")) out=corr_sorted;
        by _NAME_;
run;

proc transpose data=corr_sorted out=corr_long name=ColumnVar;
```

```
        by _NAME_;
run;

/* Step 3: Create a heatmap of the correlations
red = high correlation
white = mid correlation
blue = low correlation */
proc sgplot data=corr_long noautolegend;
        heatmapparm x=ColumnVar y=_NAME_ colorresponse=COL1 /
        colormodel=(blue white red)
        outline;
        xaxis discreteorder=data display=(nolabel);
        yaxis discreteorder=data display=(nolabel);
        title "Spearman Correlation Matrix (Numeric Variables)";
run;
```

bivariate_tests.sas

```
/*spearman (corrélation non paramétrique) */
proc corr data=games spearman;
        var _numeric_;
run;

/*Test de Kruskal-Wallis (quantitative vs catégorielle)*/
proc npar1way data=games wilcoxon edf;
        class estimated_owners;      /* variable catégorielle */
        var _numeric_;          /* variable quantitative */
run;

/*Test de Kruskal-Wallis (quantitative vs catégorielle)*/
proc npar1way data=games wilcoxon edf;
        class rating_levels;              /* variable catégorielle */
```

```
	var _numeric_;		/* variable quantitative */
run;



/*Test de normalité*/
proc univariate data=games normal;
	var _numeric_;
run;
```

univ_rating.sas

```
proc freq data=games;
	tables rating_levels / out=rating_freq;
run;

proc sgplot data=rating_freq;
	vbar rating_levels / response=percent stat=sum datalabel;
	yaxis label="Proportion (%)";
	xaxis discreteorder=data;
	title "Proportion of Game Ratings";
run;
```

lm_and_hypotheses.sas

```
/*Résidus vs valeurs ajustées, normalité, homoscédasticité, influence */
proc reg data=games;
	model average_playtime_forever =  peak_ccu price recommendations required_age
positive negative /
	vif
	r
	influence
	dwProb
```

```
        collin; /* collin pour détection de multicolinéarité */
        output out=reg_out
        r=residus
        student=student_res
        cookd=cook_dist
        h=lev
        p=valeurs_ajustees;
run;

/*QQ-plot (normalité des résidus)*/
proc univariate data=reg_out normal;
        var residus;
        qqplot residus / normal(mu=est sigma=est);
run;

/*Homoscédasticité : résidus vs ajustés*/
proc sgplot data=reg_out;
        scatter x=valeurs_ajustees y=residus;
        refline 0 / axis=y lineattrs=(pattern=shortdash);
run;

/*Studentized residuals, leverage, Cook's distance*/
proc sgplot data=reg_out;
        scatter x=lev y=student_res;
run;

data reg_out_num;
        set reg_out;
        obs_id = _N_;
run;
```

```
proc sql noprint;
        select count(*) into :n_obs from reg_out;
quit;

proc sgplot data=reg_out_num;
        scatter x=obs_id y=cook_dist;
        refline %sysevalf(4 / &n_obs) / axis=y lineattrs=(color=red thickness=2
pattern=shortdash);
        title "Cook's Distance per Observation";
run;
```

lm_selection.sas

```
proc glmselect data=games outdesign=design;
  class estimated_owners; /*variables catégorielles*/
  model average_playtime_forever = estimated_owners peak_ccu price recommendations
required_age positive negative / selection=none;
run;

/*combinaisons de variables de niveau 1 */
proc glmselect data=games;
  model average_playtime_forever = peak_ccu price recommendations required_age
positive negative
        / selection=forward(select=aic) details=all;
run;


/* Sélection pas à pas avec AIC et jeu de validation (30 %) */
proc glmselect data=games;
  partition fraction(validate=0.3); /* 30 % des données pour la validation */
  model average_playtime_forever = peak_ccu price recommendations required_age
positive negative
        / selection=stepwise(select=aic choose=validate stop=none) details=all;
```

```
run;
```

classification.sas

```
/* Log-transform and clip features */
data games_log;
	set games;
	log_positive = min(log(1 + positive), 100);
	log_peak_ccu = min(log(1 + peak_ccu), 100);
run;

/* Standardize*/
proc standard data=games_log mean=0 std=1 out=games_scaled;
	var log_positive log_peak_ccu;
run;

/* Multinomial logistic regression */
proc logistic data=games_scaled;
	class estimated_owners (param=ref);
	model estimated_owners = log_positive log_peak_ccu / link=glogit;
	output out=logit_out predprobs=I;
run;

/* VIF to check multicollinearity */
proc reg data=games_scaled;
	model log_positive = log_peak_ccu / vif;
run;

proc reg data=games_scaled;
	model log_peak_ccu = log_positive / vif;
run;
```

```
/* Generate predicted class */
data pred_vs_true;
	set logit_out;
	pred_class = I_estimated_owners;
run;

/* Confusion matrix */
proc freq data=pred_vs_true;
	tables estimated_owners * pred_class / nopercent norow nocol;
run;

/* Compute accuracy */
data accuracy;
	set pred_vs_true;
	correct = (estimated_owners = pred_class);
run;

proc means data=accuracy mean;
	var correct;
run;
```